N° d'ordre

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR DE TROISIEME CYCLE

"Mathématiques Appliquées"

par

Karen de POLIGNAC

—o—

UTILISATION DU LANGAGE CASSANDRE

POUR LA CONCEPTION DES MACHINES MICROPROGRAMMEES

—o—

Thèse soutenue le 16 Juin 1973 devant la commission d'examen

President : Monsieur J. Kuntzmann

Examinateurs : Messieurs L. Bolliet

J. Mermet

N° d'ordre

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR DE TROISIEME CYCLE

"Mathématiques Appliquées"

par

Karen de POLIGNAC

—o—

UTILISATION DU LANGAGE CASSANDRE

POUR LA CONCEPTION DES MACHINES MICROPROGRAMMEES

—o—

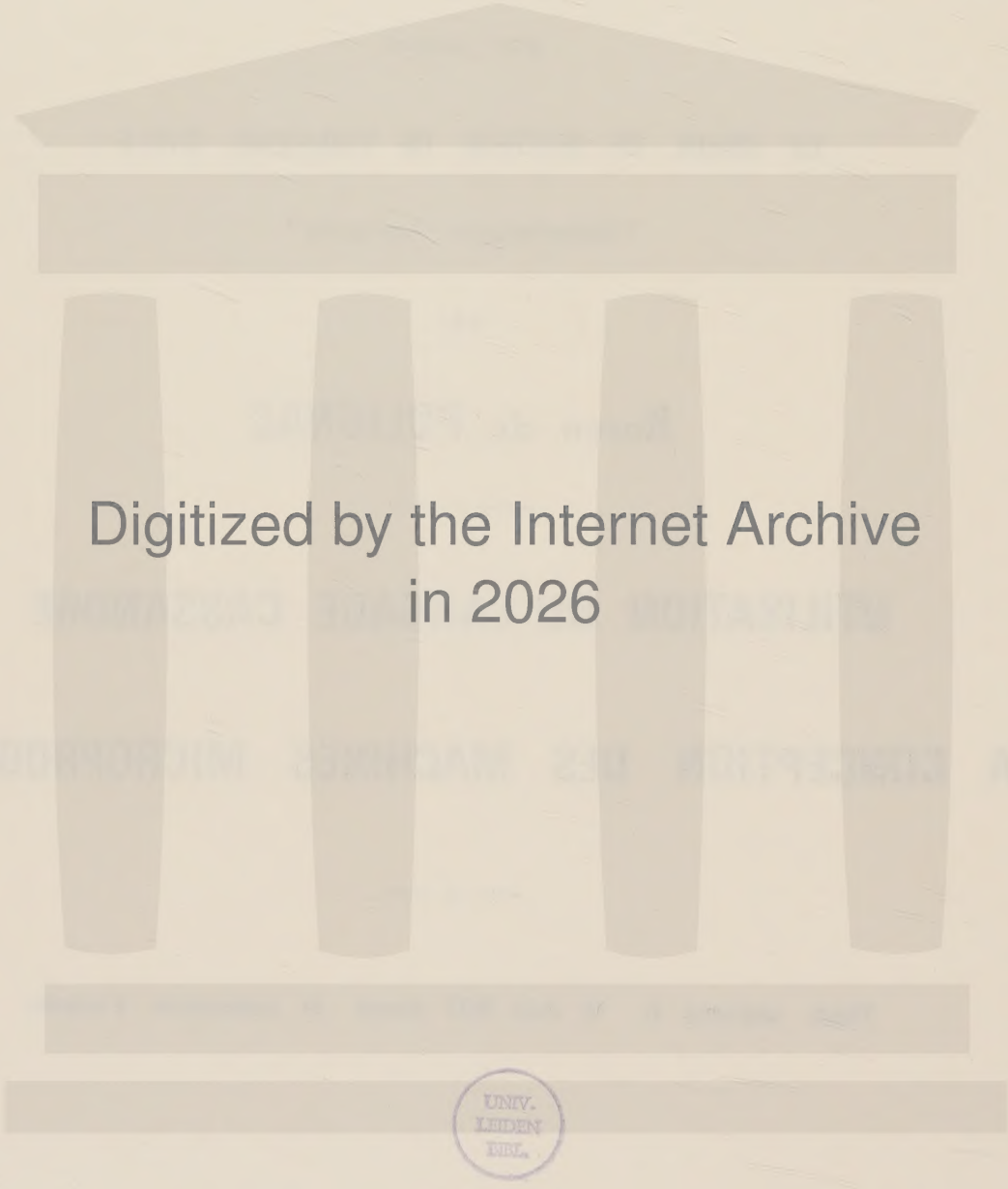
Thèse soutenue le 16 Juin 1973 devant la commission d'examen

President : Monsieur J. Kuntzmann

Examinateurs : Messieurs L. Bolliet

J. Mermet

Digitized by the Internet Archive
in 2026



UNIV.
LEIDEN
BIBL.

https://archive.org/details/IA41544319_0048

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

LISTE DES PROFESSEURS

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	ARNAUD Paul	Chimie
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BENOIT Jean	Radioélectricité
	BERNARD Alain	Mathématiques Pures
	BESSON Jean	Electrochimie
	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BONNIER Etienne	Electrochimie Electrométallurgie
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BRAVARD Yves	Géographie
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	CABANAC Jean	Pathologie chirurgicale
	CABANEL Jean	Clinique rhumatologique et hydrologie
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Oto-Rhino-Laryngologie
	CHATEAU Robert	Thérapeutique
	CHENE Marcel	Chimie papetière
	COEUR André	Pharmacie chimique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique

Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie
	FAU René	Clinique neuro-psychiatrique
	FELICI Noël	Electrostatique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GIRAUD Pierre	Géologie
	KLEIN Joseph	Mathématiques Pures
Mme	KOFLER Lucie	Botanique et Physilogie végétale
MM.	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MASSEPORT Jean	Géographie
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEEL Louis	Physique du solide
	OZENDA Paul	Botanique
	PAUTHENET René	Electrotechnique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET René	Servomécanismes
	PILLET Emile	Physique industrielle
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	SANTON Lucien	Mécanique
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SILBERT Robert	Mécanique des fluides
	SOUTIF Michel	Physique générale

MM.	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLAND François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
M.	VERAIN André	Physique
Mme	VEYRET Germaine	Géographie
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	BULLEMER Bernhard	Physique
	HANO JUN-ICHI	Mathématiques Pures
	STEPHENS Michaël	Mathématiques appliquées

PROFESSEURS SANS CHAIRE

MM.	BEAUDOING André	Pédiatrie
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BONNETAIN Lucien	Chimie minérale
Mme	BONNIER Jane	Chimie générale
MM.	CARLIER Georges	Biologie végétale
	COHEN Joseph	Electrotechnique
	COUMES André	Radioélectricité
	DEPASSEL Roger	Mécanique des fluides
	DEPORTES Charles	Chimie minérale
	GAUTHIER Yves	Sciences biologiques
	GAVEND Michel	Pharmacologie
	GERMAIN Jean-Pierre	Mécanique
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	HACQUES Gérard	Calcul numérique
	JANIN Bernard	Géographie
Mme	KAHANE Josette	Physique
MM.	MULLER Jean-Michel	Thérapeutique
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	POULOUJADOFF Michel	Electrotechnique
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	ROBERT André	Chimie papetière
	DE ROUGEMONT Jacques	Neurochirurgie
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIBILLE Robert	Construction mécanique
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBLARD Pierre	Dermatologie
	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Yves	Chimie
	BEGUIN Claude	Chimie organique
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
	BILLET Jean	Géographie
	BLIMAN Samuel	Electronique (EIE)
	BLOCH Daniel	Electrotechnique
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BOUCHET Yves	Anatomie
	BOUVARD Maurice	Mécanique des fluides
	BRODEAU François	Mathématiques (IUT B)
	BRUGEL Lucien	Energétique
	BUISSON Roger	Physique
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CHIBON Pierre	Biologie animale
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	DURAND Francis	Métallurgie
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GRIFFITHS Michaël	Mathématiques appliquées
	GROULADE Joseph	Biochimie médicale
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	JOLY Jean-René	Mathématiques Pures
	JOUBERT Jean-Claude	Physique du solide
	JULLIEN Pierre	Mathématiques Pures
	KAHANE André	Physique générale
	KUHN Gérard	Physique
	LACOUME Jean-Louis	Physique
Mme	LAJZEROWICZ Jeannine	Physique
MM.	LANCIA Roland	Physique atomique
	LE JUNTER Noël	Electronique
	LEROY Philippe	Mathématiques
	LOISEAUX Jean-Marie	Physique nucléaire
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LUU DUC Cuong	Chimie organique
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et Médecine préventive
	MARECHAL Jean	Mécanique
	MARTIN-BOUYER Michel	Chimie (CUS)

MM.	MAYNARD Roger	Physique du solide
	MICHOULIER Jean	Physique (IUT A)
	MICOUD Max	Maladies infectieuses
	MOREAU René	Hydraulique (INP)
	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PERRIN Louis	Pathologie expérimentale
	PFISTER Jean-Claude	Physique du solide
	PHELIP Xavier	Rhumatologie
Mle	RIERY Yvette	Biologie animale
MM.	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RENAUD Maurice	Chimie
	RICHARD Lucien	Botanique
Mme	RINAUDO Marquerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	SHOM Jean-Claude	Chimie générale
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VAN CUTSEM Bernard	Mathématiques appliquées
	VEILLON Gérard	Mathématiques appliquées (INP)
	VIALON Pierre	Géologie
	VOOG Robert	Médecine interne
	VROUSSOS Constantin	Radiologie
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES ASSOCIES

MM.	BOUDOURIS Georges	Radioélectricité
	CHEEKE John	Thermodynamique
	GOLDSCHMIDT Hubert	Mathématiques
	SIDNEY STUARD	Mathématiques Pures
	YACOUD Mahmoud	Médecine légale

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

Mme	BERIEL Hélène	Physilogie
Mme	RENAUDET Jacqueline	Microbiologie

Fait le 30 mai 1972.

Le présent travail a été réalisé dans le cadre d'un contrat passé entre la Direction des Recherches et Moyens d'Essais (Délégation Ministérielle pour l'armement).

Contrat n° 69.34.707.00.480.75.01

Monsieur le Professeur KUNTZMANN, Directeur de l'ENSIMAG, en acceptant de présider le Jury me fait un grand honneur auquel je suis particulièrement sensible.

Ses conseils et ses remarques ont permis l'achèvement de ce travail et je l'en remercie vivement.

Ma gratitude ira également à :

Monsieur L. BOLLIET, Professeur à l'Institut Universitaire de Technologie de Grenoble,

Monsieur J. MERMET, Attaché de Recherches au C.N.R.S,

qui ont bien voulu accepter de faire partie du Jury.

Parmi les membres de l'équipe CASSANDRE, je voudrais particulièrement remercier Monsieur J. MERMET qui m'a fait connaître le langage, Monsieur F. ANCEAU qui a suggéré l'idée de cette thèse et Messieurs A. HANCZAKOWSKI et G. MENARD qui ont participé à l'élaboration de certains programmes.

Je ne saurais, enfin, oublier dans mes remerciements Madame NEUMANN pour sa frappe soignée et le service tirage qui a assuré la réalisation de ce document.

TABLE DES MATIERES

Pages

INTRODUCTION

PREMIERE PARTIE

CHAPITRE I - ETUDE DES DIFFERENTS NIVEAUX D'EQUIVALENCE 1

1 - Equivalence des variables 2

2 - Equivalence des expressions 3

a) Propriétés des opérateurs 4

α - Opérateurs de restructutation et de transformation formelle 4

β - Opérateurs arithmétiques 7

γ - Opérateurs logiques 8

δ - Opérateurs de synchronisation 10

b) Propriétés des opérandes variables généralisées 10

3 - Equivalence des instructions 12

a) Equivalence entre instructions de même genre 14

α - Règle générale 14

β - Remarques sur l'ordre syntaxique 14

γ - Remarques sur les instructions pour 15

δ - Remarques sur l'équivalence d'ensembles d'instructions de même genre 16

b) Equivalence entre instructions de genre différent 17

α - Equivalence entre instructions simples 17

β - Equivalence entre instructions simples et instructions composées 19

γ - Equivalence entre instructions composées 20

CHAPITRE II - FORMES TYPES 23

1 - Introduction des formes types 23

2 - Principe des formes types 23

a) Niveau variable 24

b) Niveau expression 25

c) Niveau instruction 25

3 - Mode de représentation interne 26

a) Généralités sur les listes 27

b) Listes et mise sous forme type 27

α - Traitements formels 28

β - Construction de la forme type 28

c) Listes et comparaison 28

4 - Description des formes types 29

a) Forme type des variables 30

b) Forme type des expressions 31

α - Expressions d'état - EXPDS 32

β - Expressions booléennes - EXPB et EXPBE 33

γ - Expressions arithmétiques - EXPA 33

δ - Expressions logiques et expressions d'horloges - EXP et EXPH 35

c) Forme type apparaissant dans les expressions et dans les instructions 37

d) Forme type des instructions - INST 37

CHAPITRE III - TRAITEMENT D'UNE UNITE CASSANDRE 39

1 - Description d'une unité 39

2 - Liste correspondant à une instruction CASSANDRE 39

a) Traitement des variables 40

b) Traitement des expressions 41

α - Algorithme de formation de la structure d'une expression 42

β - Exemple 49

c) Traitement des instructions 53

3 - Construction des formes types 56

a) Stockage des informations 57

b) Construction de la première forme type élémentaire 60

c) Passage à la forme type élémentaire suivante éventuelle 60

4 - Codage des formes types 61

a) Comparaison avec les instructions types déjà codées 61

α - Comparaison entre instructions 61

β - Comparaison entre expressions 62

γ - Comparaison entre variables 66

δ - Processus global de comparaison 67

b) Code associé à une instruction 72

5 - Fonction contrôle et opérative de l'unité 72

a) Fonction contrôle 72

b) Fonction opérative 72

SECONDE PARTIE

SYSTEMES DE PROGRAMMES 74

1 - Généralités 74

a) Editeur 74

b) Programme de vérification 74

α - Module de vérification syntaxique et d'analyse des déclaractions 74

β - Module de vérification dimensionnelle 74

c) Format d'une unité CASSANDRE en bibliothèque 75

2 - Génération de l'analyseur syntaxique 77

3 - Principe de fonctionnement. Utilisation des modules 77

a) Configuration générale du système 77

α - Schéma synoptique 77

β - Description 77

b) Fonctionnnement de l'ensemble 80
α - Analyseur 80
β - Module de sortie 80
γ - Module de minimisation 81

EXEMPLES ET RESULTATS 82

1 - Premier exemple : Unité CALMAT 83

2 - Deuxième exemple : Unité COMPUTE1 88

BIBLIOGRAPHIE 99

Annexe I : Carte syntaxique du langage CASSANDRE A1

Annexe II : Code interne du langage CASSANDRE A4

Annexe III : Forme parenthésée des formes types A8

Annexe IV : Programmes A12

INTRODUCTION

Le langage CASSANDRE fut conçu pour permettre la description du hardware. Il permet, entre autre, la description des machines séquentielles constituées d'une partie opérative et d'un automate de contrôle. Chaque état de l'automate de contrôle peut valider plusieurs instructions CASSANDRE qui représentent des actions simultanées, indépendantes ou non, par exemple : connexions de fils, chargement de registre, changement d'état, etc...

Dans cette application, nous utiliserons le langage CASSANDRE non pas pour décrire le hardware, mais pour d'écrire l'algorithme devant être exécuté par celui-ci sous contrôle de son microprogramme. La notion d'état sera identifiée à la notion de pas dans l'algorithme. Le langage CASSANDRE est particulièrement bien adapté à une telle description, car il contient exactement les notions souhaitées pour cet usage :

- opérateurs logiques
- traitement direct des tableaux
- parallélisme à tous les niveaux.

Etant donné la description en CASSANDRE des algorithmes devant régir le fonctionnement d'une machine microprogrammée, nous avons développé un outil permettant de la définir, c'est à dire de donner :

- la spécification du hardware souhaité
- le codage de la mémoire morte de contrôle associée.

Ces opérations devront être faites sous un étroit contrôle du manipulateur pour lequel les programmes ne seront que des outils et qui devra lui-même décider des optimisations à apporter.

La méthode consiste à définir une machine à partir des éléments utilisés pour la définition de ses microprogrammes. Transposée au niveau supérieur cela reviendrait à définir un calculateur en écrivant son software. Quoique naturelle, cette méthode peut, si l'on n'y prend garde, amener certains risques. Ne souhaiterons-nous pas un jour modifier ces algorithmes ou les étendre en agissant ainsi sur leurs besoins ? Ne voudrons-nous pas également utiliser le hardware à d'autres fins, avec d'autres algorithmes complétement différents ?

Au niveau software, cela revient à ne plus considérer tous les programmes devant passer sur une machine, mais un certain ensemble jugé représentatif, tout en ajoutant des contraintes de généralité sur le hardware obtenu. Ces mêmes remarques se transposent directement au niveau des microprogrammes.

L'étude présentée ci-après et l'ensemble des programmes en découlant permettent d'obtenir une spécification de la fonction contrôle et de la fonction opérative d'une machine microprogrammée à partir de sa description CASSANDRE.

La liste des différents types d'instructions, correspondant à la partie opérative, sera donnée sous forme d'une description CASSANDRE des circuits opératifs les représentant. Une première représentation de la mémoire morte, correspondant à la partie contrôle, sera donnée par un ensemble de micro-mots qui caractérisent chacun un état de la machine. Un micro-mot sera une suite de bits, où les bits à 1 indiqueront la présence, dans l'état considéré, d'une certaine micro-opération ou micro-commande.

On voit donc nettement apparaître la nécessité de savoir reconnaître "l'équivalence" de deux micro-commandes afin que chaque bit soit caractéristique d'une micro-opération différente. Nous avons dû chercher un compromis entre l'équivalence fonctionnelle très difficile à décider et l'équivalence syntaxique triviale. La solution retenue consiste à mettre les instructions CASSANDRE sous une forme type puis à comparer syntaxiquement les formes types obtenues entre elles. Cette solution possède l'avantage de permettre la modification du processus de mise sous forme type pour donner au programme le compromis vitesse-performance le plus adapté.

PREMIERE PARTIE

CHAPITRE I

Dans ce chapitre seront analysées et définies les différentes classes d'équivalence pratiquement utilisables entre instructions CASSANDRE.

Définition :

Nous considérerons comme "équivalentes" deux instructions CASSANDRE qui commandent des microactions provoquant la même évolution du système.

Notre but est de savoir reconnaître la présence de deux instructions CASSANDRE équivalentes dans deux ou plusieurs états différents de la description d'une unité. Une telle description se compose, en effet, de trois parties :

- premièrement une liste de déclarations et de spécifications,
- deuxièmement un ensemble d'instructions toujous valides,
- enfin une liste des états de l'unité et des instructions contrôlées par chaque état.

C'est donc volontairement que nous nous sommes restreints dans cette étude à la recherche de l'équivalence de deux instructions CASSANDRE données. On pourrait, par ailleurs, s'intéresser à un niveau supérieur à l'équivalence de deux descriptions CASSANDRE.

Exemple :

On peut indifféremment écrire :

```
S := si A alors ADD (X1, X2 ; *) sinon U ;
si B alors T := ADD (Y1, Y2 ; *) ;
ou  si A alors (Z1 := X1 ; Z2 := X2) sinon
        si B alors (Z1 := Y1 ; Z2 := Y2) ;
Z3 := ADD (Z1, Z2 ; *) ;
S := si A alors Z3 sinon U ;
si B alors T := Z3 ;
```

Ces deux descriptions ont en fait la même signification, mais la reconnaissance d'une telle équivalence impliquerait une comparaison fonctionnelle de toute ou parties des unités. Néanmoins, nous verrons que les formes

types adoptées pour la représentation des différentes instructions permettront de repérer certaines des équivalences de description.

D'un point de vue formel, une instruction CASSANDRE peut être considérée comme un ensemble de variables et de symboles de base ; l'association de certains de ces symboles de base (les opérateurs) et des variables constituent des expressions. Ces divers éléments contribueront à des niveaux différents à l'équivalence de deux instructions CASSANDRE.

1 - EQUIVALENCE DES VARIABLES

Les différentes conventions d'écriture de variables indicées autorisées en CASSANDRE conduiront à considérer diverses occurences comme représentatives, en fait, d'une même variable.

Exemple :

Soit un tenseur de dimensions (4, 28, 7) déclaré :

A (0 : 3, 1 : 28, 2 : 8)

Les occurences suivantes sont correctes et équivalentes :

A ≡ A (, ,) ≡ A (0 : 3,,2 : 8) ≡ A (0 : 3, 1 : 28, 2 : 8).

Par contre, les occurences décrites ci-après, concernent un sous-tableau de A de dimensions (4, 10, 5) :

A (0 : 3, 1 : 10, 3 : 7) ≡ A (, 1 : 10, 3 : 7).

En effet, les zones vides signifient que l'on considère les composantes correspondantes qui apparaissent dans la déclaration ; et aucune nouvelle déclaration n'a besoin d'être faite pour un sous-tableau.

Il faudra tenir compte de ces conventions d'écriture et de ces déclarations implicites lors de la recherche d'équivalence de deux variables.

Il faudra également se souvenir des équivalences de variables explicitement décrites par le concepteur. Il est, en effet, possible de désigner comme "équivalentes" deux variables, déclarées de même type, par l'opération élémentaire d'équivalence, symbolisée par "≡".

Exemple :

Registre A (0 : 18, 2 : 5), MEM (1 : 9) ;
A (0 : 8, 4) ≡ MEM (1 : 9) ;

Par la suite on pourrait, par exemple, utiliser indifféremment A (5, 4) ou MEM (6).

2 - EQUIVALENCE DES EXPRESSIONS

Pour établir l'équivalence de deux instructions, on sera généralement amené à reconnaître celle de deux expressions y figurant.

Définition :

Deux expressions seront dites "équivalentes" si elles sont égales.

Cette égalité peut se présenter dans deux cas :

- c'est la même expression,
- quelles que soient les valeurs données aux variables, les deux expressions ont la même valeur. Les procédés pratiques pour reconnaître s'il en est ainsi dépendent du système algébrique auquel appartient l'expression.

Les différentes formules représentant une même expression sont conséquences à la fois des propriétés des opérateurs qu'elles font intervenir et de la nature particulière de certaines des variables sur lesquelles portent ces opérateurs. C'est pourquoi nous devons préciser, d'une part, les propriétés propres à chaque opérateur tout en indiquant les règles de calcul pratique qui en découlent, et, d'autre part, les équivalences supplémentaires éventuellement introduites par les opérandes eux-mêmes. Nous verrons, au chapitre suivant, (Chapitre II), les différentes étapes du traitement au cours desquelles on utilisera ces propriétés et règles afin premièrement de réduire au maximum, par calcul formel, les différences d'écriture et deuxièmement de comparer les expressions ainsi tranformées.

On peut distinguer dans le langage quatre familles d'éléments de base :

- les entiers et identificateurs d'entiers,
- les grandeurs logiques,
- les grandeurs de synchronisation,
- les grandeurs d'état.

Chaque famille permet de construire à l'aide d'opérateurs particulier un genre d'expression que nous désignerons par :

- expressions arithmétiques,
- expressions logiques,
- expressions d'horloge,
- expression d'état.

a) Propriétés des opérateurs

Les propriétés de la plupart des opérateurs du langage doivent être étudiées dans le cadre du type d'expression où ils apparaissent, car la sémantique dépend essentiellement de la nature des grandeurs sur lesquelles ils portent.

Ainsi l'opérateur "+" figurant dans une expression arithmétique signifiera "addition arithmétique" des opérandes, alors qu'ils signifierait "union logique" dans une expression booléenne. De même l'opérateur "&" peut signifier concaténation ou intersection logique...

α - Opérateurs de restructuration et de transformation formelle

Ce sont les seuls opérateurs dont la signification et les propriétés sont indépendantes du type de grandeur sur laquelle ils agissent ; certains sont propres à la nature vectorielle de l'opérande.

* Les opérateurs concaténation "&" et transposition "∿" permettent de restructurer les variables. On peut ainsi, par un nombre quelconque de concaténations et de transpositions sur un certain nombre de variables, définir de nouvelles variables.

Exemple :

```
Registre A (1 : 3, 1 : 8), B (1 : 9, 0 : 3, 1 : 15),
         C (1 : 16, 0 : 7), D (1 : 32, 0 : 2) ;
∿ A & B (2 : 9, 0 : 2, 7) & C (, 1 : 3) & D
```

définit une variable de dimensions :

8 + 8 + 16 + 32 = 64 et 3.

En effet, l'opérateur concaténation permet d'accoler suivant la première direction deux variables ayant le même nombre de dimensions et des dimensions compatibles dans toutes les directions autres que la première.

L'opérateur transposition permute deux directions d'une variable ayant au moins deux dimensions. Les deux directions à permuter sont précisées par une paire d'entiers figurant à la suite de l'opérateur. Toutefois, cette paire d'entiers peut être omise lorsque la transposition concerne seulement les deux premières directions.

Il a été attribué à la transposition une priorité supérieure à celle de la concaténation.

La sémantique des opérations concaténation et transposition conduit aux quatre identités suivantes :

soit - A une variable de dimension au moins égale à 2,
- B une variable quelconque,
- N1 et N2 deux entiers

alors :

$$\sim A = \sim (1, 2)\ A \qquad (1)$$

$$\sim (N1, N2)A = \sim (N2, N1)\ A \qquad (2)$$

$$\sim (N1, N2)\ (\sim (N1, N2)\ A) = A \qquad (3)$$

$$B\ \&\ (\sim (N1, N2)\ A) = B\ \&\ \sim (N1, N2)\ A \qquad (4)$$

* Les opérations de décalage et de rotation définies sur des vecteurs peuvent s'étendre à chaque dimension d'un tableau de dimensions quelconques.

Les opérateurs décalage "D" et rotation "R" sont suivis d'un entier N précisant le nombre de positions décalées à gauche si N est positif ou à droite si N est négatif. Dans le cas du décalage, ces N positions débordées sont perdues et la dimension considérée de l'être est réduite ; celui-ci peut être complété par une concaténation.

Les définitions même de ces opérateurs impliquent les identités énoncées ci-dessous :

Soit - A un vecteur (ou un tableau)
- N1 et N2 deux entiers

Alors :

$$D\ |N1|\ (D\ |N2|\ A) = D\ |N1 + N2|\ A \quad \text{si } N1 + N2 > 0 \qquad (5)$$

$$R\ |N1|\ (R\ |N2|\ A) = \begin{cases} R\ |N1 + N2|\ A & \text{si } N1 + N2 \neq 0 \qquad (6) \\ A & \text{si } N1 + N2 = 0 \qquad (7) \end{cases}$$

* L'opérateur de réduction "/", défini par Iverson pour des vecteurs, peut se généraliser aux tableaux de dimension quelconque en s'appliquant à leur première dimension, position en laquelle peut être ramenée n'importe quelle direction par transposition.

Etant donnée une variable vectorielle A et un opérateur logique dyadique , on définit l'opération de réduction par :

$$/\ \lambda\ A\ (0 : N) = A\ (0)\ \lambda\ (A\ (1)\ \lambda\ (\ldots(A\ (N-1)\ \lambda\ A\ (N)))\ldots)$$

Remarquons simplement la propriété suivante dans le cas particulier où λ se trouve être un opérateur commutatif et associatif :
"=", "≠", "ou", "et".

$$/\ \lambda\ (R\ |N|\ A) = /\ \lambda\ A \qquad (8)$$

* A l'aide des différents opérateurs décrits dans ce paragraphe, il est possible de définir de nouvelles variables, dites variables généralisées ; mais soulignons l'importance de l'ordre dans lequel peuvent apparaître plusieurs de ces opérateurs portant sur une même variable.

Exemple : Soit A le vecteur booléen suivant :

A = (0 1 0 0 1 1 0)
R |3| (D |2| A) = (1 0 0 0 1)
alors que D |2| (R |3| A) = (1 0 0 1 0)

Par application des règles (1), (2), (4), (5), (6), (7) et (8), utilisées pour remplacer, dans la description d'une variable généralisée, toute occurrence d'un groupement de gauche par celui de droite équivalent, on pourra

amener l'écriture d'une telle variable à être unique par rapport aux opérateurs de restructuration et de transformation formelle, à condition de considérer les deux entiers, intervenant après l'opérateur $\sim$, comme une paire et non comme un couple.

β - Opérateurs arithmétiques

Les expressions arithmétiques apparaissant en CASSANDRE ont comme opérandes des nombres entiers ou des identificateurs d'entiers et comme opérateurs :

+ , - , . , ./ (division entière)

Rappelons la commutativité et l'associativité de l'addition, de la soustraction et du produit, ainsi que la distributivité du produit par rapport à la somme algébrique.

Par contre, l'opération division entière ne présente aucune propriété particulière propre ou relative aux trois autres opérations.

Les identités utilisées pour la transformation formelle d'une expression arithmétique seront les suivantes :

Soient

- A, B et C des monômes dont les facteurs sont des identificateurs d'entiers ;
- α, β, γ et δ des nombres entiers.

Alors

$\alpha . A + \beta . A = \gamma . A$ avec $\gamma = \alpha + \beta$

$\alpha . A . \beta . A = \delta . A . A$ avec $\delta = \alpha . \beta$

$A . (B + C) = A . B + A . C$

Remarquons qu'après application de la distributivité et regroupement éventuel des facteurs numériques, aucune règle autre que la commutativité ne permet de transformer un produit de facteurs puisque l'opérateur puissance n'apparaît pas dans la syntaxe du langage.

γ - Opérateurs logiques

- Opérateurs booléens

Les expressions logiques apparaissant en CASSANDRE ont comme opérandes des constantes booléennes, des variables logiques simples (de type registre, mémoire ou signal) ou généralisées et comme opérateurs, les opérateurs classiques de l'algèbre de Boole :

. + (union logique), . (intersection logique), - (négation).

L'union et l'intersection sont commutatives, associatives et distributives l'une par rapport à l'autre.

La loi de Demorgan, la loi d'absorption et les règles de simplification d'un monôme ou polynôme booléen seront utilisées pour les transformations formelles éventuelles des expressions logiques.

Soient A et B deux variables booléennes ayant les mêmes dimensions :

Loi de Demorgan
$$\begin{cases} -(A \,.\, B) = (-A) + (-B) \\ -(A + B) = (-A) \,.\, (-B) \end{cases}$$

Double négation
$$-(-A) = A$$

Loi d'absorption
$$A + A \,.\, B = A$$

Simplification de monômes
$$\begin{cases} A \,.\, A = A \\ A \,.\, (-A) = 0 \\ A \,.\, 1 = A \\ A \,.\, 0 = 0 \end{cases}$$

Simplification de polynômes
$$\begin{cases} A + A = A \\ A + (-A) = 1 \\ A + 1 = 1 \\ A + 0 = A \end{cases}$$

0 (ou 1) désignant une constante booléenne de mêmes dimensions que A et B et dont toutes les composantes sont égales à 0 (ou à 1).

Notons les propriétés particulières de l'opérateur négation "-", relativement aux opérateurs définis au paragraphe α.

A, B et C étant des variables booléennes vectorielles :

- (A & B & C) = (-A) & (-B) & (-C)
- (∿ (N1, N2) A) = ∿ (N1, N2) (-A)
- (R |N| A) = R |N| (-A)

Notons également la distributivité de l'opération concaténation par rapport aux opérations d'union et d'intersection logique :

- A & (B + C + D) = A & B + A & C + A & D
- A & (B . C) = A & B . A & C

Enfin, les opérations union et intersection logiques étant étendues à des tableaux de dimensions quelconques par calcul composante à composante, il y a aussi "distributivité" des opérateurs transposition et rotation par rapport à celles-ci. La "distributivité" d'un opérateur s'entendant de la façon suivante :

- ∿ (N1, N2) (A + B) = ∿ (N1, N2) A + ∿ (N1, N2) B
- ∿ (N1, N2) (A . B) = ∿ (N1, N2) A . ∿ (N1, N2) B
- R |N| (A + B) = R |N| A + R |N| B
- R |N| (A . B) = R |N| A . R |N| B

- Opérateurs de comparaison et de relation

Ces opérateurs permettent de représenter en CASSANDRE certaines fonctions booléennes de deux variables, ce sont :

$=$, $\neq$, $>$, $\geq$, $\leq$, $<$

Les fonctions booléennes associées à ces opérateurs sont les suivantes :

$A = B$	$\Longleftrightarrow$	$\bar{A} \cdot \bar{B} + A \cdot B$
$A \neq B$		$A \cdot \bar{B} + \bar{A} \cdot B$
$A < B$		$\bar{A} \cdot B$
$A > B$		$A \cdot \bar{B}$
$A \leq B$		$\bar{A} + B$
$A \geq B$		$A + \bar{B}$

Lors de la description d'une unité, le concepteur aura pu utiliser indifféremment l'une ou l'autre de ces deux formes et il faudra en tenir compte au moment de la comparaison de deux opérandes d'expressions logiques.

δ - Opérateurs de synchronisation

Les opérateurs unaires dérivation "δ" et retard "τ" permettent de décrire la synchronisation des signaux.

L'opérateur dérivation, propre aux expressions d'horloge permet de passer d'un signal à une impulsion.

L'opérateur retard associé à une durée exprimée par un nombre entier permet de retarder des variables de type signal ou impulsion et peut donc apparaître dans une expression logique ou d'état.

Ces deux opérateurs unaires ne présentent aucune propriété propre ou relative aux autres opérateurs rencontrés dans les expressions construites à partir des grandeurs logiques ou des grandeurs de synchronisation.

Les expressions, dites d'horloge, peuvent donc définir de nouvelles horloges par union "+" et concaténation "&" d'horloges existantes ou créées par dérivation de grandeurs logiques.

Soulignons la distributivité de la concaténation par rapport à l'union d'impulsions.

(H1 + H2) & H = H1 & H + H2 & H

b) Propriétés des opérandes variables généralisées

Nous avons vu que nous pouvons définir de nouvelles variables par concaténation de deux ou plusieurs grandeurs logiques ou grandeurs de synchronisation. C'est ce caractère "concaténé" éventuel des variables généralisées qui introduit des possibilités supplémentaires d'équivalence dans l'ensemble des expressions logiques comme dans l'ensemble des expressions d'horloge.

Exemple 1 : Soient les variables définies ci-après :

C ; L (1 : 2) ; I, B, F, D, M (1 : 3) ; K (1 : 4) ;
J, E (1 : 5) ; A (1 : 8)

et soient deux expressions E1 et E2 construites sur ces variables :

E1 = A&B&C&D&E . I&J&F&C&D&L&M . A&F&K&E
E2 = I&J&B&C&D&L&M . A&B&K&E . I&J&F&C&D&E . A&F&C&D&L&M

Si nous considérons chacun des facteurs de ces monômes comme une entité, on peut aussi écrire :

E1 = X1 . X2 . X3 en posant X1 = A&B&C&D&E
X2 = I&J&F&C&D&L&M
X3 = A&F&K&E

et

E2 = X4 . X5 . X6 . X7 en posant X4 = I&J&B&C&D&L&M
X5 = A&B&K&E
X6 = I&J&F&C&D&E
X7 = A&F&C&D&L&M

car les sept facteurs apparaissant dans les expressions sont distincts. Alors, les deux monômes, portant sur des facteurs différents, ne peuvent être égaux.

Par contre, si nous considérons les éléments concaténés constituant les facteurs des monômes et le résultat de l'intersection logique composante à composante, telle qu'elle doit effectivement être évaluée, nous verrons que ce résultat est le même pour E1 et pour E2 :

A(1) . I(1) & A(2) . I(2) & A(3) . I(3) & A(4) . J(1) & A(5) . J(2)
& A(6) . J(3) & A(7) . J(4) & A(8) . J(5) & B(1) . F(1) & B(2) . F(2)
& B(3) . F(3) & C . K(1) & D(1) . K(2) & D(2) . K(3) & D(3) . K(4)
& E(1) . L(1) & E(2) . L(2) & E(3) . M(1) & E(4) . M(2) & E(5) . M(3)

Exemple 2 : Considérons le développement de l'expression suivante :

(K & C & B + I & J) & (A + E & F + L & D & M)
= K & C & B & A + K & C & B & E & F + K & C & B & L & D & M
+I & J & A + I & J & E & F + I & J & L & D & M

Dans cette expression développée il y a redondance de termes. En effet, elle pourrait, par exemple, être représentée par seulement 3 termes (termes soulignés).

Aucune formule, aucune identité, ne permet de traduire une équivalence comme celle que nous avons pu constater entre E1 et E2, ni de donner des règles propres à la simplification d'une expression portant sur des variables construites par concaténation. Néanmoins, nous verrons qu'à la phase de comparaison des expressions, un traitement tenant compte des dimensions de chacun des éléments concaténés permettra de reconnaître de telles équivalences.

3 - EQUIVALENCE DES INSTRUCTIONS

Pour l'étude des équivalences à ce niveau, il est utile de diviser l'ensemble des instructions CASSANDRE en deux sous-ensembles : d'une part, celui des instructions dites "simples" et d'autre part, celui des instructions dites "composées". Les instructions du second groupe se distinguent de celles du premier par le fait, qu'au cours de leur analyse, on rencontre d'autres instructions.

Instructions simples

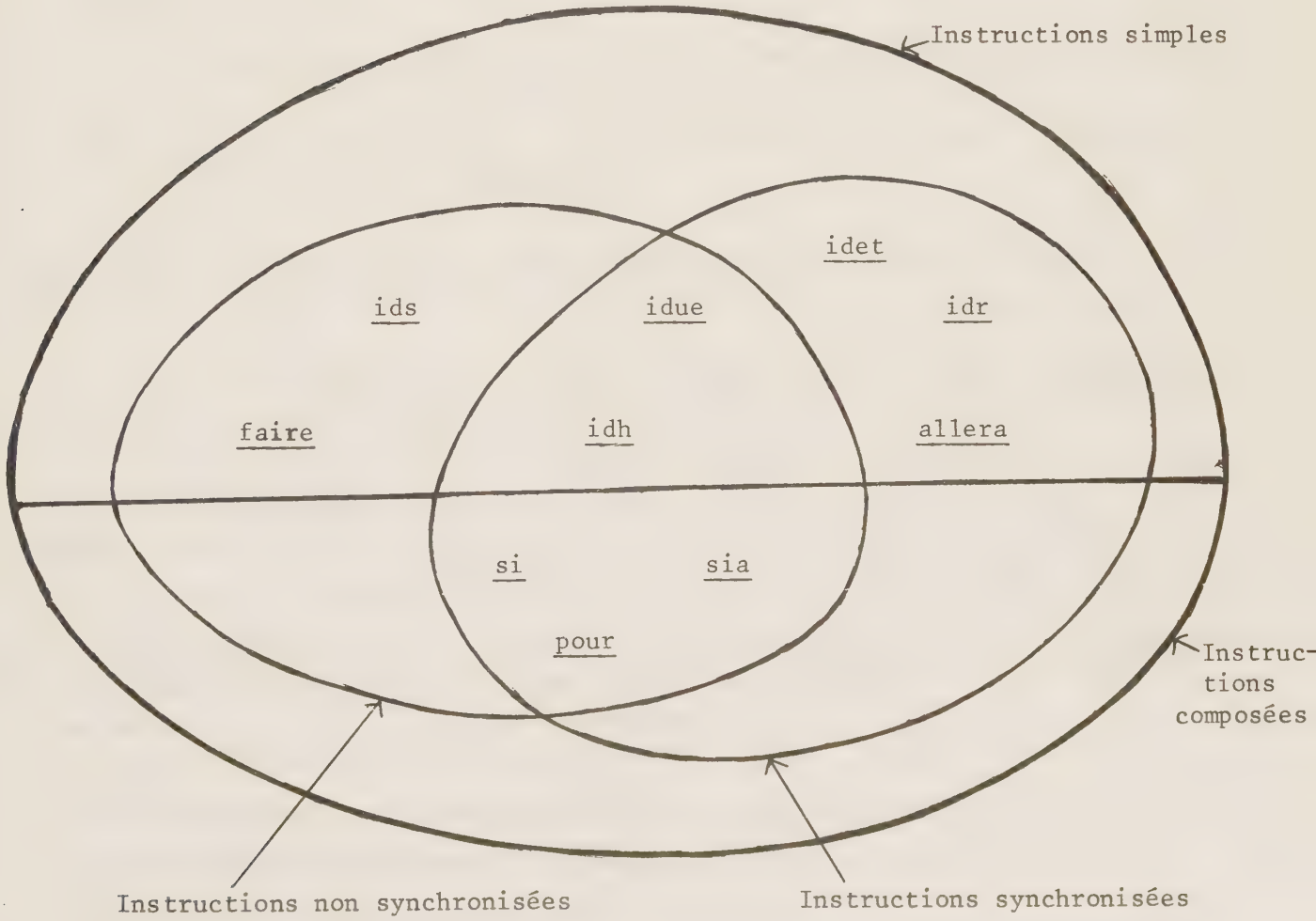
	- Connexion de signal	ids
(H)	- Connexion d'unité	idue
(H)	- Génération d'impulsion	idh
H	- Chargement de variable d'état	idet
H	- Chargement de registre	idr
H	- Ordre allera	allera
	- Ordre faire	faire

Instructions composées

(H)	- Instruction conditionnelle	si
(H)	- Instruction arithmétique conditionnelle	sia
(H)	- Boucle pour	pour

A l'intérieur de chaque groupe on peut encore distinguer différentes sortes d'instructions (voir aussi figure ci-dessous) : celles qui doivent être synchronisées par une impulsion (instructions étiquettées H), celles qui ne dépendent jamais d'une impulsion de synchronisation (instructions sans étiquettes) et enfin celles qui, suivant les cas peuvent, ou non, dépendre d'une telle impulsion (instructions étiquettées (H)).

Pour désigner plus simplement par la suite tous ces différents **genres** d'instructions, nous utiliserons le symbole de base CASSANDRE par lequel elles doivent commencer : par exemple, idh (identificateur d'horloge), pour la génération d'impulsion ...



Partition de l'ensemble des instructions du langage CASSANDRE

a) Equivalence entre instructions de même genre

α - Règle générale

Une première recherche d'équivalence peut se faire à l'intérieur d'un même genre d'instructions. Elle conduira à la reconnaissance de l'identité des symboles de base et à la détermination de l'équivalence des variables, des expressions et aussi, éventuellement, des instructions qui figurent dans les deux instructions examinées, les divers éléments que l'on sera amené à comparer apparaissant dans un ordre syntaxiquement fixé.

β - Remarques sur l'ordre syntaxique

* Il faut noter, puisqu'aucune notion d'enchaînement séquentiel n'existe entre les instructions CASSANDRE, la commutativité :

- des instructions sur lesquelles peut porter une boucle "pour".

Exemple : pour A = 2 à 10 début I1 ; I2 ; ... ; In fin ;
commutativité

- des instructions sur lesquelles peut porter une instruction arithmétique conditionnelle.

Exemple : sia A > 2 alors début I1 ; I2 ; ... ; In fin ;
commutativité

- des instructions sur lesquelles peut porter une instruction conditionnelle dont la condition est une variable booléenne scalaire.

Exemple : si C alors début I1 ; I2 ; ... ; In fin sinon début J1;J2;...;Jp fin
commutativité commutativité

- des instructions constituant l'ensemble des actions conditionnées par une composante du tableau résumant les conditions logiques d'une instruction conditionnelle généralisée.

Exemple : si I (1 : 3) alors (I1;I2;...;In)(J1;J2;...;Jp)(L1;L2;...;Lq) ;

commutativité

- des opérations conditionnées par une impulsion dans une instruction d'affectation.

Exemple : <H> I1, I2, ..., In ;

commutativité

* Une autre exception à la rigueur de l'ordre syntaxique est fournie par les instructions conditionnelles pures ou arithmétiques. En effet, deux instructions de ce type peuvent être équivalentes si les portées des "alors" et des "sinon" sont inversées et si les conditions sont la négation logique l'une de l'autre.

Exemple : Soient I et J deux blocs d'instructions :

si A + (-B) alors I sinon J ;

est évidemment équivalent à :

si B . (-A) alors J sinon I ;

γ - Remarques sur les instructions pour

La recherche d'équivalence entre instructions pour présente plusieurs particularités par rapport à la description générale qui en a été faite en début de paragraphe : test, dans un ordre donné, de l'identité ou de l'équivalence des éléments constituant les deux instructions considérées.

* Variables contrôlée. La variable contôlée par une boucle pour étant une variable muette, l'identificateur qui la représente peut être différent dans deux instructions pour qui sont néanmoins équivalentes.

Exemple : pour I = 1 à N début A(I) := B(I+1, I) ; fin ;
pour K = 1 à N début A(K) := B(K+1, K) ; fin ;

* Bornes de la variables contrôlée. Lorsque la variable contrôlée ne figure pas explicitement dans la portée de la boucle pour, ce n'est pas la valeur

propre de ses bornes qui a une signification, mais seulement la valeur de leur différence, celle-ci indiquant le nombre de fois que l'on désire exécuter l'instruction sur laquelle porte la boucle.

Exemple : pour I = 1 à 7 début A := B + C . A ; fin ;
pour N = S à S + 6 début A := B + C . A ; fin ;

* Imbrication de boucles pour. Une affectation placée à l'intérieur d'une boucle pour ou, plus généralement, d'un ensemble de boucles pour imbriquées, correspond à deux sortes de calculs différents : soit une évaluation de certains éléments de vecteur, tableau ou tenseur, soit un calcul de récurrence. Seulement dans le premier cas, l'ordre d'imbrication des boucles est quelconque.

Exemples :

(1) pour I = 1 à 4 début
pour J = 0 à N début
pour K = 5 à J + 7 début
A(I, J, K) := B(I - J + K) ; fin ; fin ; fin ;

est équivalent à :

pour J = 0 à N début
pour K = 5 à J + 7 début
pour I = 1 à 4 début
A(I, J, K) := B(I - J + K) ; fin ; fin ; fin ;

(2) Par contre, A ayant été initialisé :

pour I = 0 à 3 début
pour J = 1 à 2 début A := I + J . A ; fin ; fin ;

n'est pas équivalent à

pour J = 1 à 2 début
pour I = 0 à 3 début A := I + J . A ; fin ; fin ;

δ - Remarques sur l'équivalence d'ensembles d'instructions de même genre

Nous avons, jusqu'à présent, seulement envisagé la possibilité d'équivalence instruction à instruction. Or, la définition même des instructions composées dont la portée est une autre instruction et, plus généralement,

un bloc d'instructions, conduit à considérer aussi des équivalences entre ensembles d'instructions composées, certains ensembles pouvant se réduire à une seule instruction.

Exemple : pour N = 1 à 3 début I1 ; I2 ; I3 ; I4 ; I5 ; fin ;

Cette instruction est équivalente à l'ensemble suivant, que nous écrivons ici regroupé, mais qui n'au aucune raison particulère de l'être dans un programme CASSANDRE.

pour N = 1 à 3 début I2 ; I5 ; fin
⋮
pour N = 1 à 3 début I4 ; fin ;
⋮
pour N = 1 à 3 début I3 ; I2 ; fin ;

I1 faut encore signaler les équivalences particulières introduites par les instructions conditionnelles (arithmétiques ou non) parmi les équivalences entre ensembles d'instructions composées. Ces équivalences résultent à la fois de la remarque générale faite plus haut et de celle faite en β) sur ces mêmes instructions.

Exemple :

sia A = 1 alors début I1 ; I2 ; fin ;
⋮
sia A ≠ 1 alors début I3 ; fin sinon début I4 ; I5 ; fin ;

Cet ensemble est équivalent à :

sia A = 1 alors début I4 ; fin sinon I3 ; fin ;
⋮
sia A = 1 alors début I1 ; fin ;
⋮
sia A = 1 alors début I5 ; I2 ; fin ;

b) Equivalence entre instructions de genre différent

α - Equivalence entre instructions simples

I1 y a peu d'équivalences possibles entre instructions simples de genre différent, et elles sont toutes dûes à la notion, particulière au langage CASSANDRE, de "signal-unité".

Outre la connexion d'unité simple, résumant, en fait, tout un ensemble de connexions de signaux et d'horloges, qui sont les entrées ou les sorties de l'unité considérée, il existe une connexion d'unité, dite "signal-unité", dans laquelle l'une des entrées ou sorties est remplacée par le signe *.

La présence d'un "signal-unité" est équivalente à celle de la variable (signal ou horloge) d'entrée ou sortie remplacée par l'étoile. On pourra donc, d'une part, employer un "signal-unité" dans une expression lorsque l'étoile remplace une sortie et, d'autre part, l'employer comme partie gauche d'un branchement lorsque l'étoile remplace une entrée.

La notion de "signal-unité" peut conduire à considérer comme équivalentes premièrement, des instructions non sous horloge de genre connexion d'unité et d'autres de genre connexion de signal ou génération d'impulsion, et deuxièmement, des instructions sous horloge de genre connexion d'unité et d'autres de genre chargement de registre ou génération d'impulsion. Il faut préciser, à ce niveau, que dans les genres connexion de signal et génération d'impulsion sont comprises des instructions qui commencent, en fait, par le symbole de base CASSANDRE idue (emploi d'un signal unité comme partie gauche de branchement).

Exemples d'équivalences entre instructions simples.

```
Unité ADDER (E(1 : 8), H ; S(0 : 5)) ;
Horlogemère H ; Horloge H1 ;
Externe N12 ((1 : 1), (1 : 1) ; (1 : 1)) ;
        N13 ((1 : 1), (1 : 1), (1 : 1) ; (1 : 1)) ;
        N14 ((1 : 1), (1 : 1), (1 : 1), (1 : 1) ; (1 : 1)) ;
```

Les instructions suivantes de connexion d'unité :

```
N12 |1| (E(1), E(2) ; S(1)) ;
N14 |2| (S(3), S(1), E(7),  E(8) ; S(4)) ;
N13 |1| (E(1),,E(3) ; H1) ;
```

sont équivalentes aux instructions de connexion de signal et de génération d'impulsion décrites ci-après :

```
S(1) := N12 |1| (E(1), E(2) ; *) ;
N14 |2| (S(3), S(1), *, E(8) ; S(4)) := E(7) ;
H1 := N13 |1| (E(1),,E(3) ; *) ;
```

β - Equivalence entre instructions simples et instructions composées

Par analyse de la syntaxe des instructions simples, on constate qu'il y figure toujours au moins une expression et, en particulier, que celle-ci peut être arithmétique.

Or, le langage comprend pour chaque genre d'expressions : logique, arithmétique, d'horloge et d'état, et, des expressions simples et des expressions conditionnelles. Les conditions des expressions arithmétiques conditionnelles étant de même nature que celles figurant dans les instructions conditionnelles arithmétiques (sia), alors que les conditions des autres expressions conditionnelles sont de même nature que celles des instructions conditionnelles (si).

Donc, puisque toute instruction simple peut porter sur une expression conditionnelle, arithmétique ou autre, elle peut aussi être équivalente à une instruction composée de genre si ou sia.

Nous pouvons préciser ces équivalences en tenant compte du fait qu'une instruction, simple ou composée, peut être, ou non, conditionnée par une impulsion de synchronisation.

Il est évident qu'il peut y avoir équivalence entre instruction simple et instruction composée conditionnelle de même caractère quant au conditionnement par une horloge. En outre, toute instruction simple sous horloge peut être équivalente à une instruction composée si ou sia non sous horloge. En effet, du fait même que l'instruction soit composée, l'impulsion peut n'être mentionnée qu'à un niveau plus interne.

Exemples :

(1) - Les instructions décrites ci-dessous, respectivement de genre faire et de genre si sont équivalentes :

- faire si C alors ETAO sinon ETA2 ;
- si C alors (faire ETAO) sinon (faire ETA2) ;

(2) - Ce second exemple montre l'équivalence d'une instruction H-idr avec une instruction H-sia et avec une instruction sia

```
- <H>   R <= S (sia A > 2 alors 3 sinon B) ;
- <H>   sia A > 2 alors début R <= S(3) ; fin
                  sinon début R <= S(B) ; fin ;
- sia A > 2 alors début <H>   R <= S(3) ; fin
            sinon début <H>   R <= S(B) ; fin ;
```

γ - Equivalence entre instructions composées

D'une part, la remarque faite précédemment sur le niveau quelconque auquel peut apparaître l'impulsion dans une instruction composée permet une équivalence éventuelle entre instructions composées de même genre, l'une sous horloge et l'autre non.

D'autre part, une instruction sia comme une instruction pour peut porter sur n'importe quel genre d'instruction et en particulier l'une sur l'autre, d'où une équivalence possible, que ces instructions dépendent ou non d'une impulsion.

Exemples :

(1) - Equivalence entre instructions de même genre sous horloge et non sous horloge :

```
- <H>   pour I = 1 à 5 début H1 (I) := A (I-2) ; fin ;
- pour I = 1 à 5 début <H>   H1 (I) := A (I-2) ; fin ;
```

(2) - Equivalence entre instructions composées de genre différent :

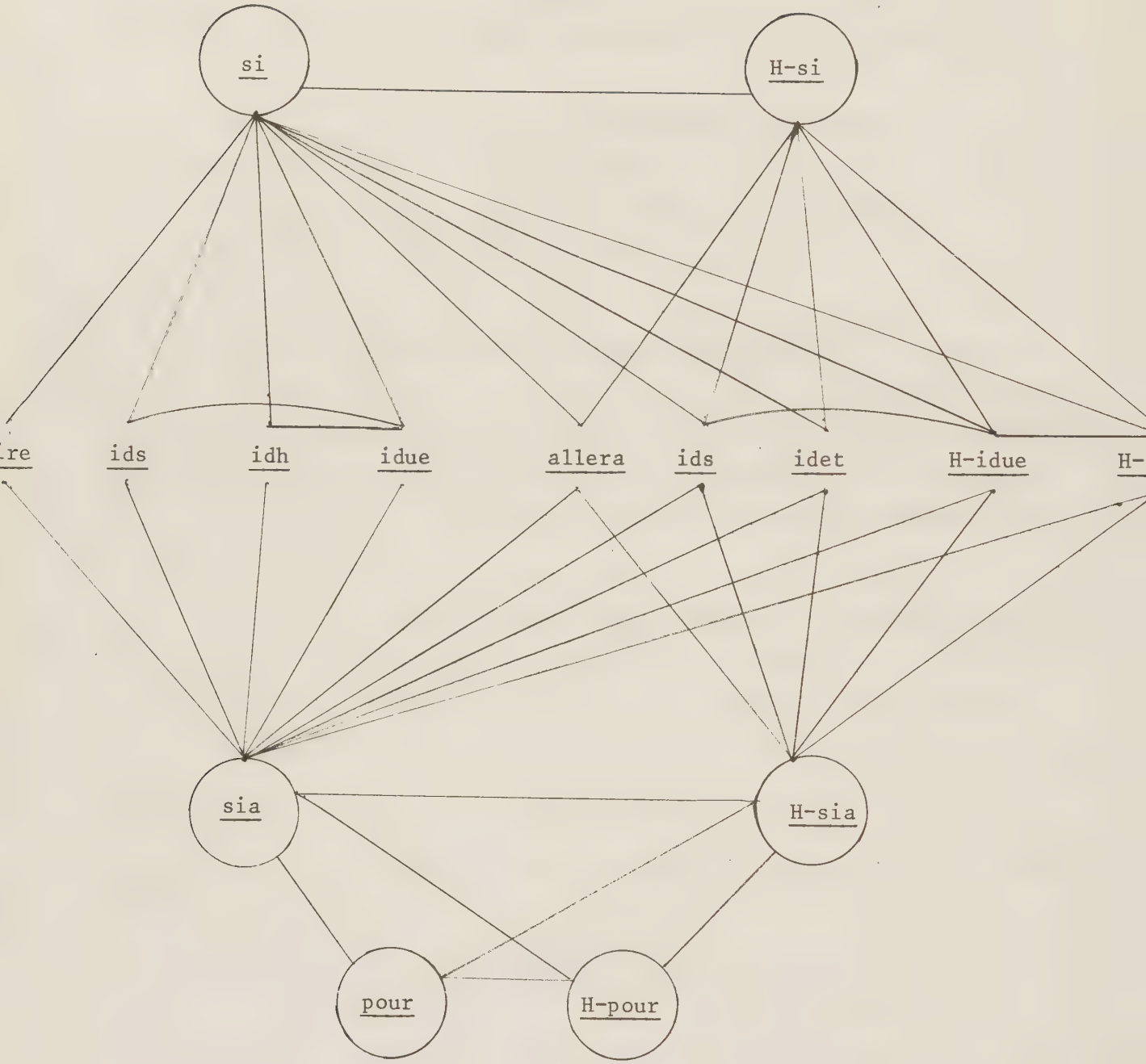
```
- sia B > 2 alors début pour I = 1 à 5
                        début <H>   H1 (I) := A (I-2) ; fin ; fin ;
- pour I = 1 à 5 début sia B > 2 alors
                       début <H>   H1 (I) := A (I-2) ; fin ; fin ;
- <H>   sia B > 2 alors début pour I = 1 à 5
                             début H1 (I) := A (I-2) ; fin ; fin ;
```

- <H> pour I = 1 à 5 début sia B > 2 alors
début H1 (I) := A (I-2) ; fin ; fin ;

Rappelons que les instructions sia et pour de cet exemple sont aussi équivalentes aux instructions de même genre :

- sia B > 2 alors début <H> pour I = 1 à 5
début H1 (I) := A (I-2) ; fin ; fin ;
- pour I = 1 à 5 début <H> sia B > 2 alors
début H1 (I) := A (I-2) ; fin ; fin ;

Le graphe de la page 22 résume toutes les possibilités d équivalence. Deux instructions I2 et I2 sont en relation R, et liées sur le graphe si et seulement si I2 et I1 peuvent être équivalentes. Toutefois, afin d'alléger la figure, les liaisons représentant la réflexivité de la relation R n'ont pas été mentionnées. On peut remarquer que la relation R considérée est réflexive, symétrique mais non transitive.



xxxx : instruction simple

xxxx : instruction composée

Equivalences possibles entre instructions CASSANDRE

CHAPITRE II

Au cours du chapitre précédent nous avons dégagé les principales formes d'équivalence qu'il serait intéressant de savoir reconnaître. Nous allons maintenant étudier et définir des méthodes permettant de déceler rapidement ces équivalences.

1 - INTRODUCTION DES FORMES TYPES

Il apparaît, dès à présent, que la reconnaissance de l'équivalence de deux instructions va conduire à "comparer" chaque nouvelle instruction avec toutes les instructions différentes trouvées parmi celles déjà analysées.

Ce procédé appelle deux remarques. D'une part, la "comparaison" mise en oeuvre devra être totalement différente et beaucoup plus approfondie qu'une comparaison syntaxique dont le chapitre I a souligné l'insuffisance. Nous avons vu, en effet, que les genres CASSANDRE d'instructions (ou expressions) n'étaient pas caractéristiques et que de nombreuses équivalences étaient possibles entre genres différents. D'autre part, cet algorithme de "comparaison" d'une instruction avec toutes les précédentes est, a priori, extrêmement lourd et nous avons été amenés à trouver des moyens propres à l'accélérer.

C'est pourquoi, nous avons cherché à mettre les instructions (et les expressions) sous une forme, dite "forme type" distinctive et unique (à certaines commutativités près) pour chaque catégorie et c'est sous cette forme que se fera la recherche des équivalences. Celle-ci consistera, alors, en une comparaison quasi-syntaxique d'une instruction avec celles de même type précédemment retenues.

2 - PRINCIPES DES FORMES TYPES

Les formes types adoptées ont été choisies pour satisfaire un certain nombre de critères généraux.

Nous avons recherché le nombre minimum de formes types grâce auquel il serait possible, d'une part, de représenter n'importe quelle instruction ou expression CASSANDRE et, d'autre part, de repérer toutes les équivalences mentionnées dans le chapitre précédent.

L'analyse d'une instruction CASSANDRE provoquera la construction d'une forme type ou, d'une façon plus générale, d'un ensemble de formes types lui correspondant. Cette construction se fera progressivement au cours de l'analyse. Afin de la rendre la plus rapide possible les formes types ont été définies pour rester très proches des genres CASSANDRE des instructions et expressions les plus simples et pour ne contenir qu'un minimum de renseignements tirés de la chaîne CASSANDRE.

Les formes types ont été introduites dans le but d'accélérer le processus de comparaison et les choix précédemment faits pour une élaboration rapide des formes types répondent aussi à ce besoin. Il s'avère donc intéressant de mettre la chaîne analysée sous une forme telle que la position d'une information représente elle-même un renseignement (par exemple : le niveau auquel elle se situe) et que seuls soient conservés les symboles de base caractérisant le type (par exemple : tous les termes d'une expression polynomiale seront situés à un même niveau et on pourra supprimer, sans créer d'ambiguité, les opérateurs "+" qui les séparent).

Enfin les formes types devront être caractéristiques et entièrement indépendantes de façon que l'on puisse à chaque niveau, abandonner la recherche d'équivalence en cas de réponse négative lors d'une comparaison.

En plus de ces principes généraux sur le choix des formes types nous avons dégagé quelques critères plus particuliers à chacun des trois niveaux CASSANDRE déjà mis en évidence : niveaux variable, expression et instruction.

a) Niveau variable

Les variables CASSANDRE seront toujours considérées comme indicées et les formes types leur correspondant comprendront deux éléments : un représentant la description du libellé et l'autre représentant la liste des indices.

Ce second élément pourra éventuellement être totalement ou partiellement vide, mais, de toutes façons, ne contiendra que les renseignements mentionnés dans la chaîne CASSANDRE analysée et non ceux que l'on pourrait tirer de la partie déclaration du programme en cas de dimension ou borne non précisée. Toutes les dimensions seront situées à un même niveau et pour chacune d'elle sera donné, quand il figure, l'indice ou la paire de borne indiqué.

Un indice ou une borne sera représenté soit par une expression arithmétique, soit par une expression logique précédée de l'opérateur $ (valeur numérique).

b) Niveau expression

Les six genres différents d'expression du langage pourront être ramenés à une même forme type de base, dite forme polynomiale : c'est à dire une "somme" de termes où chaque terme est un "produit" de facteurs et les expressions se différencieront par la nature de leurs facteurs. Les mots "somme" et "produit" ne sont pas à interpréter avec leur signification habituelle, mais seulement à comprendre comme un découpage des expressions en deux niveaux principaux. Une "somme" pourra, par exemple; aussi bien être le résultat d'addition d'éléments arithmétiques, que le résultat d'union logique d'éléments booléens ou encore d'union d'éléments d'impulsion. Une forme type polynomiale comprendra donc deux niveaux : l'un terme et l'autre facteur et, par suite, ne figureront plus que les opérateurs portant directement sur les facteurs.

La forme type correspondant à une expression représentera, en fait, sa forme la plus "développée" et la plus "simplifiée".

Par exemple : 2 + I + 8 . (J+1) et I + 10 + 8 . J auront la même représentation type.

Remarquons qu'aucune forme type conditionnelle des expressions n'a été définie car nous verrons plus loin comment toutes les conditions apparaissant dans les expressions seront reportées au niveau de l'instruction.

c) Niveau instruction

C'est à ce niveau que les formes types se différencient le plus de la forme CASSANDRE car, le plus souvent, à une instruction CASSANDRE correspondra un ensemble d'instructions sous forme type.

A une instruction simple CASSANDRE où ne figure que des expressions non conditionnelles correspondra une instruction sous forme type. Nous avons reconnu 7 genres d'instructions simples et chacun sera représenté par une forme type différente.

Les instructions composées si, sia et pour portent, d'une façon générale, sur un ensemble d'instructions. La forme type correspondante retenue ne porte que sur une seule instruction. Il y aura donc génération d'autant d'instructions sous forme type si, sia ou pour qu'il en figure dans la portée du modèle CASSANDRE.

Il y aura fission des instructions si et sia en deux instruction sous forme type, si celle analysée comporte une partie sinon ; la première instruction aura comme condition celle figurant dans l'instruction CASSANDRE et la seconde la négation logique, tandis que les éléments effectifs seront respectivement les portées du alors et du sinon. En fait, il n'y aura qu'une seule forme type d'instruction conditionnelle et l'instruction si se différencie de l'instruction sia par le genre de l'expression constituant la condition.

Toutes les conditions apparaissant dans la description d'une instruction CASSANDRE, tant au niveau variable (par les expressions représentant les bornes ou indices), qu'au niveau expression proprement dit ou instruction, seront imbriquées. A chaque combinaison possible des conditions correspondra une instruction type où toute instruction ou expression conditionnelle aura été remplacée par l'élément effectif associé à la condition prise en compte.

Nous pouvons donc préciser maintenant qu'il existera 9 formes types d'instructions différentes, dont deux composées et que, pour ces dernières, l'instruction interne ne peut être qu'une instruction simple ou une instruction pour. En outre, chacune de ces formes types pourra éventuellement être conditionnée par une impulsion, celle-ci étant toujours remontée au niveau d'instruction le plus externe.

3 - MODE DE REPRESENTATION INTERNE

L'intérêt de l'utilisation des formes types étant établi, il reste à en définir le mode de représentation. Toujours avec les critères de rapidité et d'efficacité de leur construction et des comparaisons, il apparaît qu'une représentation interne par listes se prête bien, par son principe et par ses propriétés, à cette utilisation.

a) Généralités sur les listes

Une liste est constituée d'éléments qui sont des mots mémoire. Chaque élément mémoire se divise en deux parties égales :

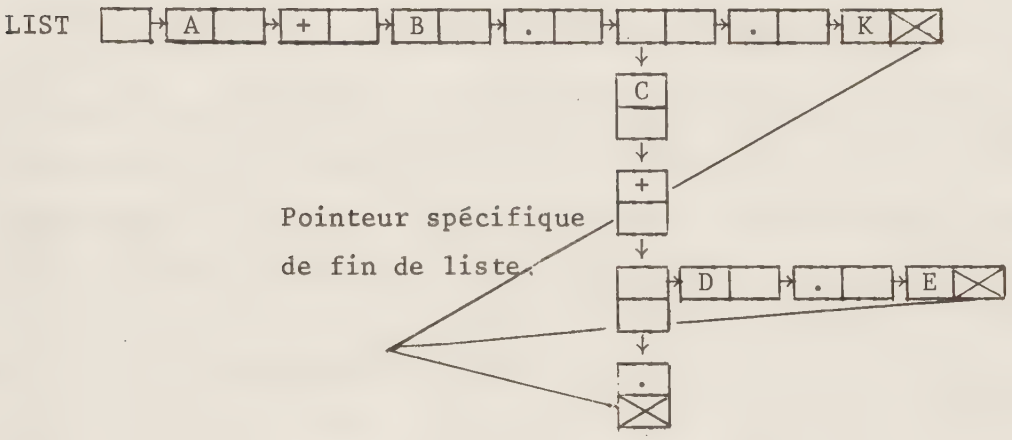
- une partie gauche qui correspond à la partie information et qui contient soit un atome, c'est à dire une information codée, soit un pointeur vers une sous-liste.

- une partie droite qui réalise le chaînage avec les autres éléments mémoire et qui contient donc un pointeur vers l'élément suivant de la liste.

Exemple d'une représentation possible par liste de la formule :

A + B . (C + (D.E)') . K

Remarque : ce n'est pas celle de la forme type correspondant à cette formule.



b) Listes et mise sous forme type

A chaque nouvelle instructions CASSANDRE analysée on fera donc correspondre une instruction (ou un ensemble d'instructions) sous forme type représentée par une liste (ou un ensemble de listes) avant de la comparer aux précédentes.

L'utilisation des listes pour la mise sous forme type présente deux aspects. Elle permet, d'une part, d'effectuer certains traitements formels et, d'autre part, de construire la forme type elle-même.

α - Traitements formels

Une instruction, ou expression, analysée peut se trouver sous une forme très différente de sa forme type classique. Nous avons vu que, d'une façon générale, la forme type n'est pas la représentation fidèle des éléments analysés, mais celle d'une instruction équivalente ou d'une expression égale. La construction d'une liste parallèlement à l'analyse facilité l'application de certains traitements aux éléments de la liste de façon à se ramener à la forme qui sera représentée par la forme type.

Au niveau de l'instruction, ce traitement peut, par exemple, consister à l'imbrication des conditions. Dans une expression ce sera essentiellement du calcul formel : mise sous forme de somme de produits, application de la loi de Demorgan à une expression logique...

β - Construction de la forme type

La liste associée à une instruction analysée est construite et modifiée au fur et à mesure de l'analyse en fonction de la forme type correspondante à réaliser. Par conséquent, quand l'analyse de l'instruction est terminée, la liste a sa structure définitive ; elle représente la forme type de l'instruction et va pouvoir être utilisée pour les comparaisons qui tiendront compte du fait que l'on a éventuellement obtenu tout un ensemble d'instructions sous forme type.

c) Listes et comparaisons

L'utilisation des listes s'avère aussi très intéressante au niveau des comparaisons. Pour comparer une nouvelle instruction à une instruction déjà trouvée, on va comparer les listes construites à partir de ces deux instructions. La représentation par listes permet, en effet, d une part, de bien distinguer les différents niveaux d'une instruction ou d'une expression et, d'autre part, de facilement tenir compte des propriétés de commutativité qui peuvent exister.

4 - DESCRIPTION DES FORMES TYPES

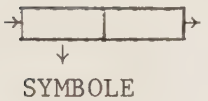
L'étude de l'ensemble des instructions et expressions autorisées par le langage CASSANDRE dans la description d'une unité a permis de définir un certain nombre de formes types que nous allons préciser dans ce paragraphe. Ces formes types seront décrites sous forme de listes puisque c'est le mode de représentation interne adopté. On distinguera ici les différents niveaux de la variable, de l'expression et de l'instruction, et pour chacun d'eux on donnera la caractérisation des formes types correspondantes.

Il est important de noter que ces formes types sont les formes définitives, c'est à dire celle obtenues après l'ensemble des traitements et utilisées pour les comparaisons.

Les conventions adoptées pour la représentation de ces formes types sont les suivantes :

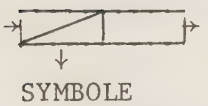
représente le code CASSANDRE (1 octet) de définition du type d'une variable dans une expression ou d'un symbole de base.

représente un pointeur (2 octets) vers la description d'une variable créée dans l'espace liste par le vérificateur.



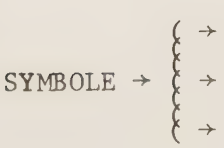
représente un pointeur vers une sous-liste précisée par SYMBOLE qui renvoie à une forme type décrite ailleurs sous ce nom.

indique que l'élément d'information attendu à cet endroit est inexistant ou implicite.



signifie que l'on peut avoir l'un ou l'autre des deux cas précédents.

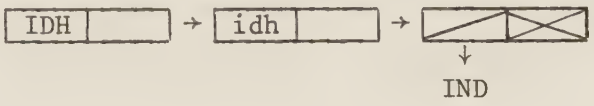
la partie droite de cet élément est le marqueur fin de liste.



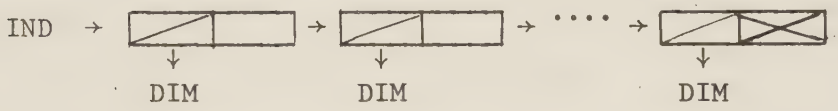
la forme type repérée par le nom SYMBOLE peut avoir l'une ou l'autre des configurations indiquées.

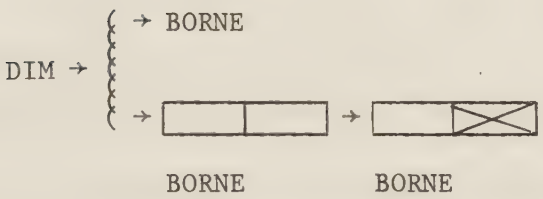
a) Formes types des variables

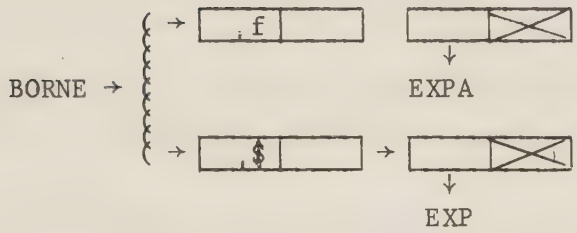
Une variable CASSANDRE n'est soumise à aucun traitement. Elle sera simplement représentée par un pointeur vers sa description dans l'espace liste précédé, si nécessaire, du code qui définit son type et suivi d'une liste d'indices.

Exemple : 

La forme type d'une liste d'indices est la suivante :







Le symbole de base "f" n'appartient pas au langage CASSANDRE. Il a été introduit ici artificiellement de façon à maintenir une symétrie entre les deux formes possibles de BORNE.

Une variable au sens large se compose d'une grandeur élémentaire -variable simple ou indicée ou expression non conditionnelle- précédée d'un ou plusieurs opérateurs rangés dans un ordre déterminé.

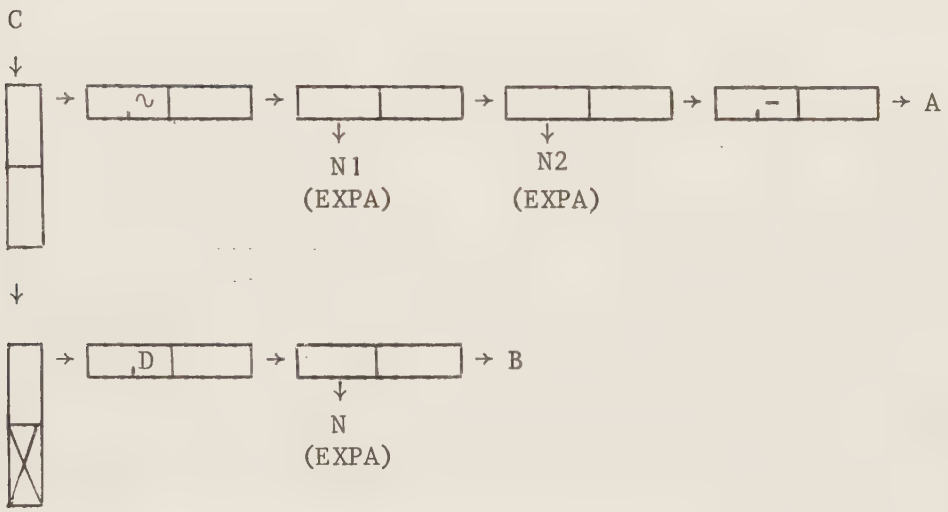
Les différents opérateurs pouvant apparaître dans une variable au sens large sont :

/λ, δ, ∿ (N1, N2), R |N|, D |N|, -, τ |N|.

Une variable généralisée s'obtient par la concaténation de variables au sens large. L'exemple ci-dessous montre la représentation de ces deux types de variables.

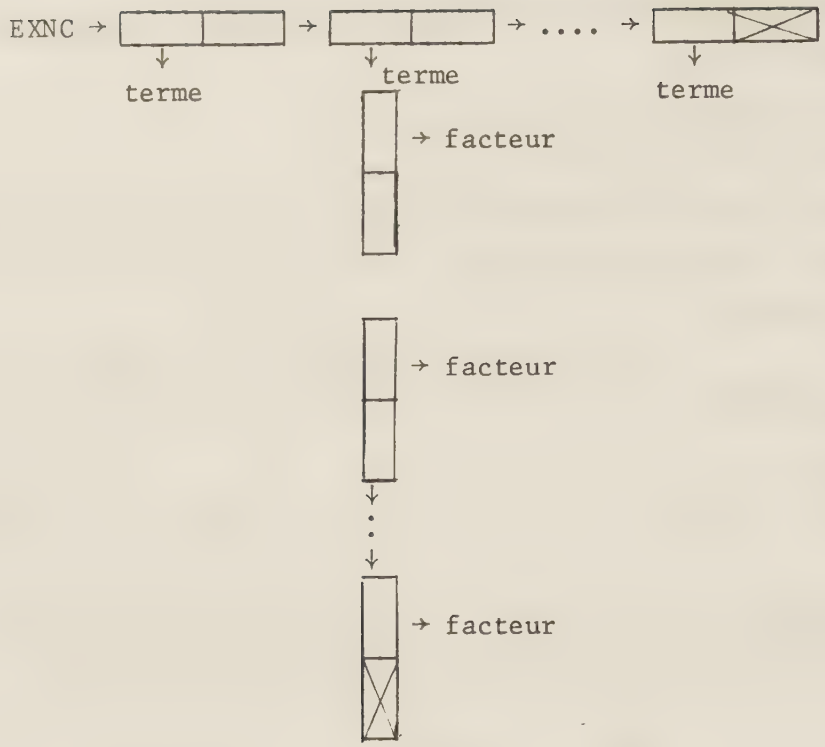
registre A (1 : 3, 1 : 10), B (0 : 6) ;

C := ∿ (N1, N2) - A & D |N| B ;



b) Formes types des expressions

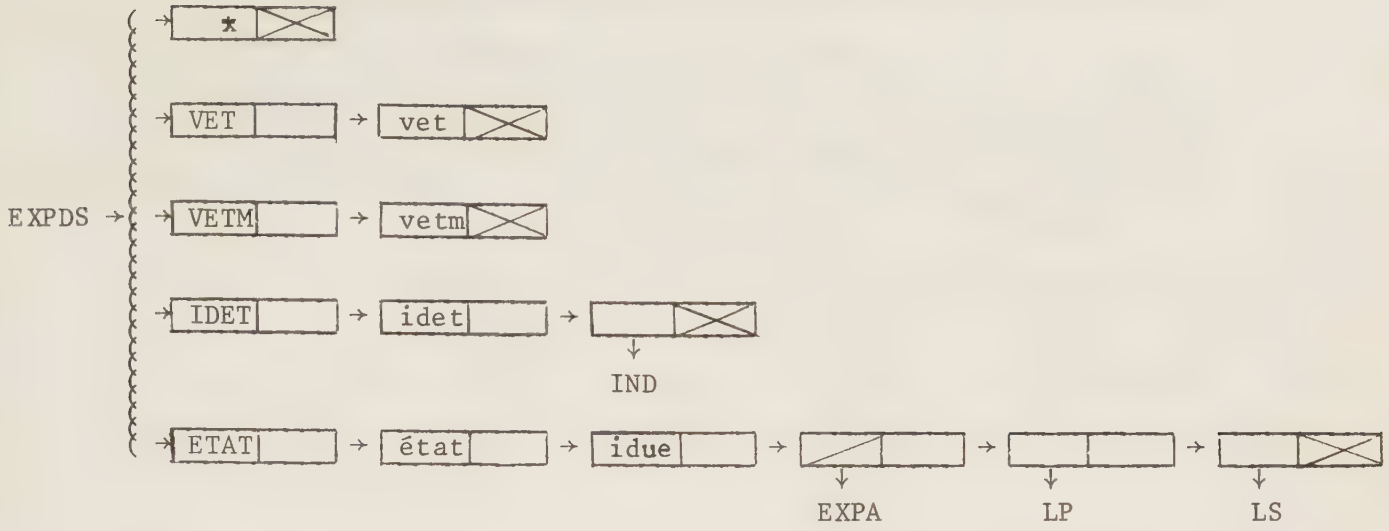
Une seule forme type générale a été retenue pour les expressions : la forme non conditionnelle, dite polynomiale. Cette forme a deux niveaux principaux : les sous-listes pointées à un premier niveau représenteront les termes et celles pointées au niveau inférieur représenteront les facteurs. On sait alors que, selon la nature de l'expression, il faut faire la somme ou l'union logique des termes et le produit ou l'intersection des facteurs.



Il reste maintenant à préciser la forme type des facteurs correspondant à chacun des différents genres d'expressions CASSANDRE. On peut remarquer qu'aucun renseignement indiquant le genre de l'expression ne figure dans les formes types. En effet, l'analyse, la construction et les comparaisons seront conduites de façon à ce que l'on connaisse à tout moment le genre de l'expression traitée.

α - Expression d'état - EXPDS

Elle a pour résultat une variable ou valeur d'état et, à cause même de sa simplicité, sera la seule expression non mise sous forme polynômiale mais représentée par une valeur immédiate d'état pouvant avoir l'une des configurations décrites ci-après :



β - Expressions booléennes - EXPB et EXPBE

Ce sont des polynômes dont les facteurs sont des grandeurs booléennes "vrai" ou "faux" résultant de la comparaison de deux expressions :

- arithmétiques pour EXPB
- d'état pour EXPBE.

Ces deux classes d'expressions seront donc toujours représentées sous forme polynômiale, la configuration des facteurs étant la suivante :

. facteur type de EXPB : PPB → OR → → (EXPA) → (EXPA)

OR = opérateur de relation $\{>, \geq, =, \neq\}$

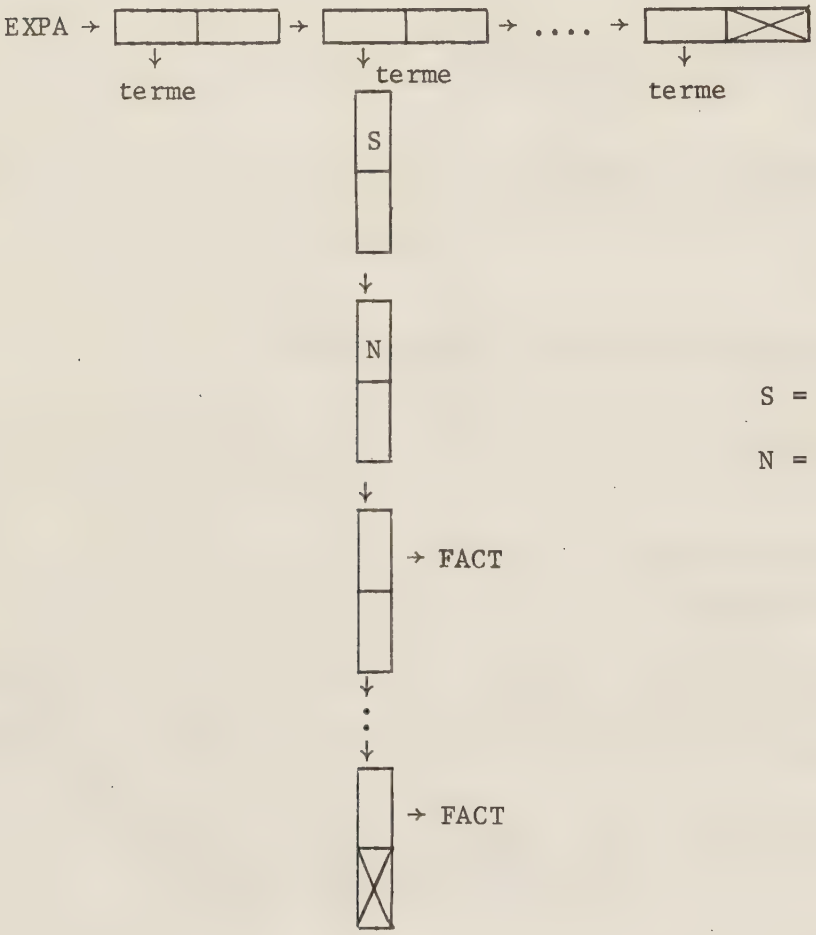
. facteur type de EXPBE : PPBE → OC → → (EXPDS) → (EXPDS)

OC = opérateur de comparaison $\{=, \neq\}$

γ - Expressions arithmétiques - EXPA

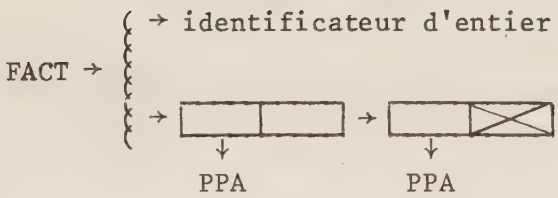
Elles portent sur des grandeurs arithmétiques entières : entiers identificateurs d'entiers, ou résultats de divisions entières. Chaque terme

du polynôme forme type comprend deux facteurs particuliers : l'un représentant le signe du terme considéré et l'autre regroupant tous les facteurs numériques. Les autres facteurs constituant le terme seront donc, soit un pointeur vers la description d'un identificateur d'entier, soit un pointeur vers une sous-liste représentant une division entière dont on précisera les deux opérandes.



S = signe {+, -}

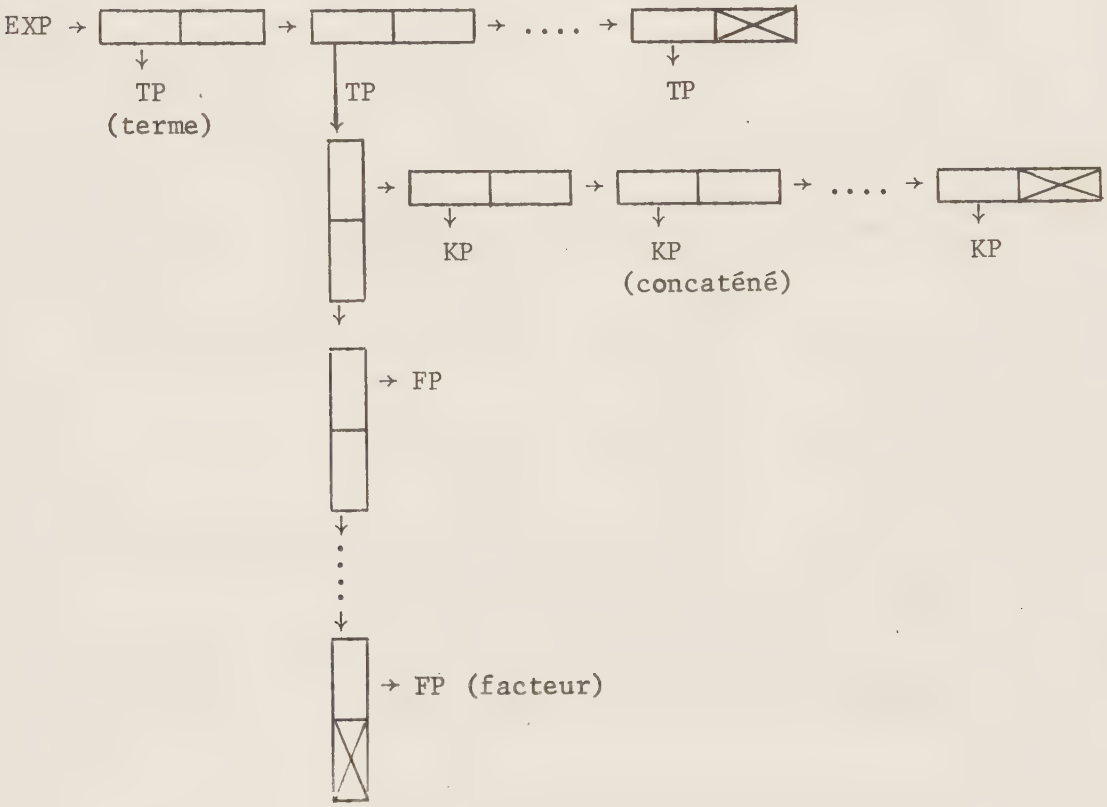
N = entier {1, ...}

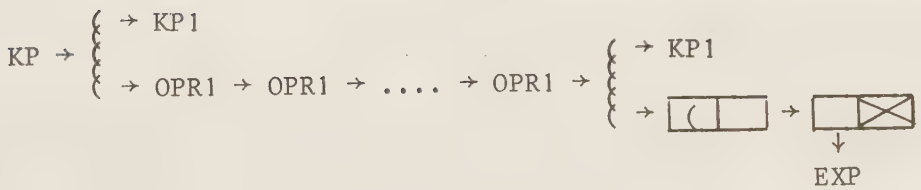


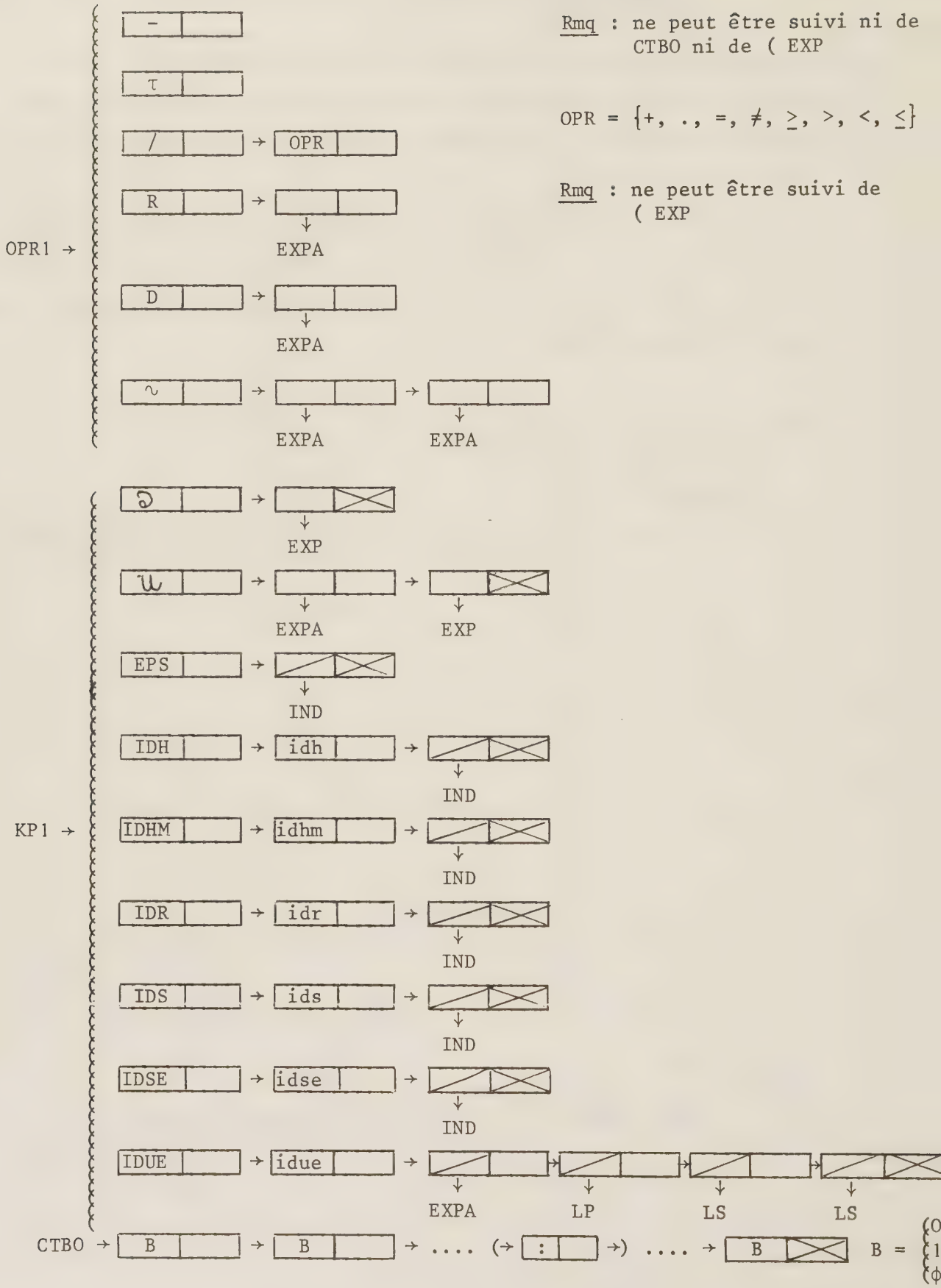
PPA → { → identificateur d'entier
→ entier
→ EXPA

δ - Expressions logiques et expressions d'horloge - EXP et EXPH

En regroupant les représentations des grandeurs logiques et des grandeurs de synchronisation, il est possible de ne donner qu'une seule description pour ces deux genres d'expressions. Pour l'une, comme pour l'autre, les grandeurs considérées étant des variables généralisées, c'est à dire des concaténations de variables au sens large, il a été nécessaire de créer un niveau supplémentaire pour la représentation des concaténés. On peut remarquer que seules les opérations d'union et de concaténation d'horloges sont définies dans le langage, si bien que les termes du polynôme forme type d'une expression d'horloge n'auront jamais qu'un seul facteur.

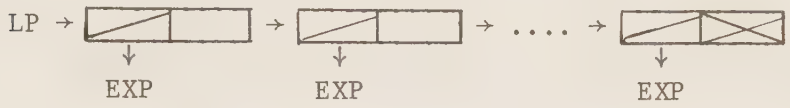


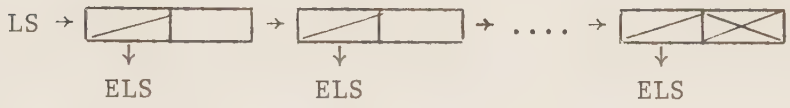


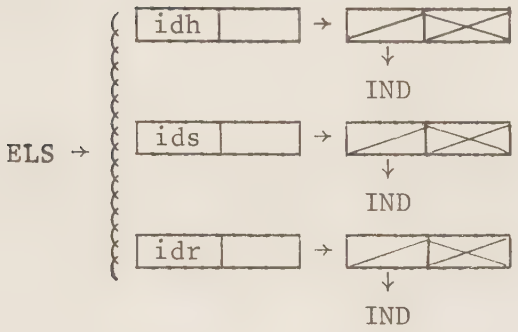

Rmq : ne peut être suivi ni de CTBO ni de (EXP
OPR = {+, ., =, ≠, ≥, >, <, ≤}
Rmq : ne peut être suivi de (EXP
OPR1 →
-
τ
/
OPR
R
EXPA
D
EXPA
∿
EXPA
EXPA
∂
EXP
ω
EXPA
EXP
EPS
IND
IDH
idh
IND
KP1 →
IDHM
idhm
IND
IDR
idr
IND
IDS
ids
IND
IDSE
idse
IND
IDUE
idue
EXPA
LP
LS
LS
CTBO →
B
B
.... (→ : →) ....
B
B =

c) Formes types apparaissant dans les expressions et dans les instructions

Ce sont celles des listes d'entrées et de sorties des unités externes utilisées comme signal-unité ou dans une connexion d'unité. Elles comportent autant d'éléments que d'entrées ou sorties déclarées pour l'unité, mais seules celles précisées lors de l'utilisation sont décrites par la forme type.

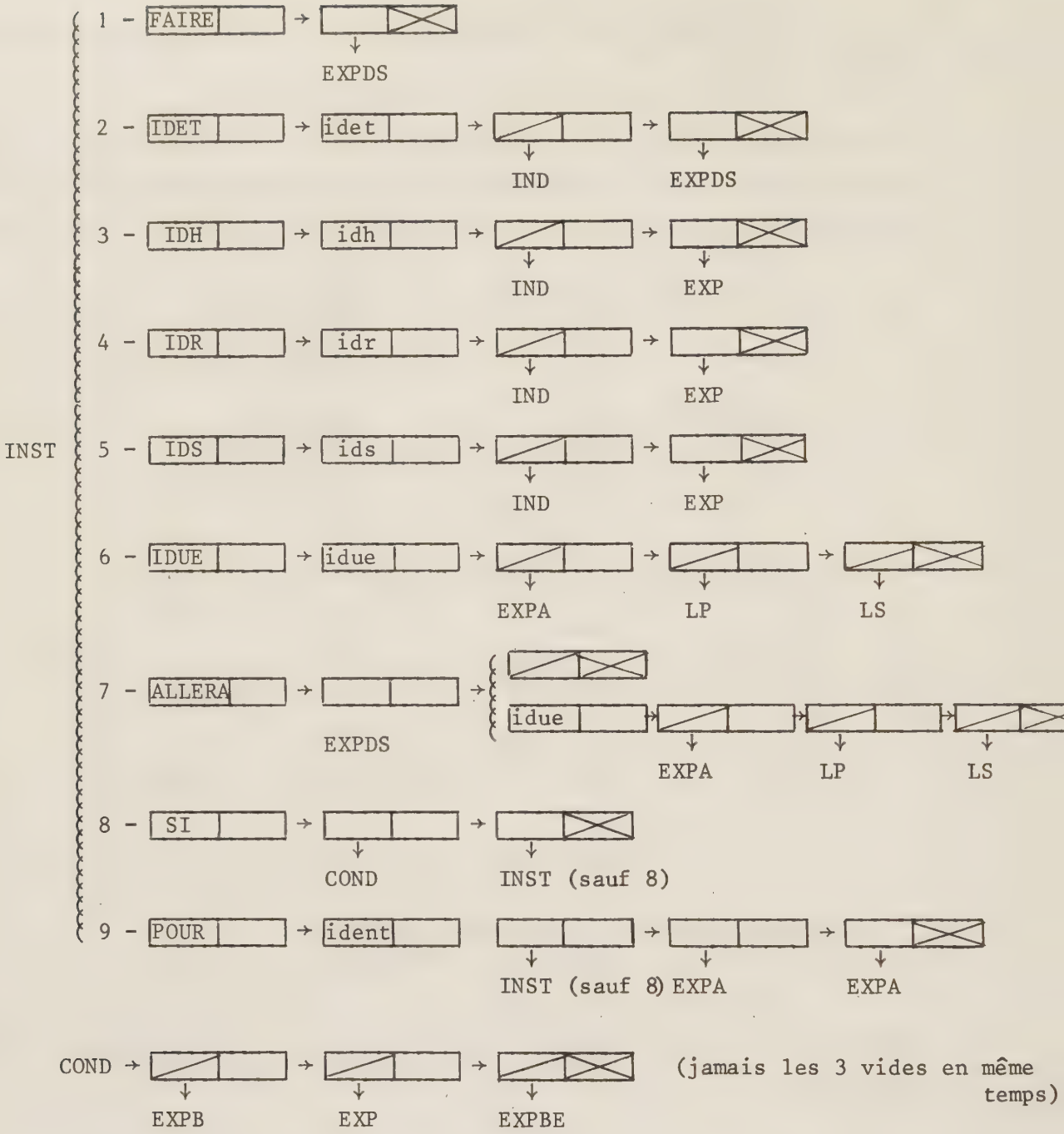






d) Formes types des instructions - INST

Nous avons vu qu'elles étaient au nombre de neuf, dont deux seulement composées et que pour celles-ci l'instruction interne ne devait pas être conditionnelle.


INST
1 - FAIRE
EXPDS
2 - IDET
idet
IND
EXPDS
3 - IDH
idh
IND
EXP
4 - IDR
idr
IND
EXP
5 - IDS
ids
IND
EXP
6 - IDUE
idue
EXPA
LP
LS
7 - ALLERA
EXPDS
idue
EXPA
LP
LS
8 - SI
COND
INST (sauf 8)
9 - POUR
ident
INST (sauf 8)
EXPA
EXPA
COND
EXPB
EXP
EXPBE
(jamais les 3 vides en même temps)

CHAPITRE III

Le traitement d'une unité CASSANDRE, ou machine microprogrammée, doit fournir simultanément une image de la mémoire morte et une description des différentes instructions rencontrées.

1 - DESCRIPTION D'UNE UNITE

Elle se décompose en trois parties :

- déclarations et spécifications
- instructions toujours valides
- liste des états de l'unité et des instructions contrôlées par chaque état.

La première partie contenant les déclarations, ne sera l'objet d'aucun traitement. Elle a, en effet, été prise en compte par le vérificateur qui en a tiré une description des variables rangée dans l'espace liste. C'est pourquoi, dans la chaîne codée issue du vérificateur, les variables sont représentées d'abord par un code qui en définit le type, puis par un pointeur vers leur description dans l'espace liste.

Le traitement d'une unité se ramène donc à celui des instructions qui la décrivent, c'est à dire les instructions toujours valides et celles validées par chacun des états.

Ces différentes instructions sont prises en compte et traitées successivement dans l'ordre où elles se présentent, les intructions toujours valides pouvant être considérées comme validées par tous les états.

2 - LISTE CORRESPONDANT A UNE INSTRUCTION CASSANDRE

La construction des formes types se décompose en deux phases. Une première liste est construite au fur et à mesure de l'analyse d'une instruction CASSANDRE et c'est à partir de cette liste que sera générée la liste (ou l'ensemble des listes) représentant la forme type (ou les formes types) correspondant à l'instruction analysée.

Nous allons donner pour chacun des trois niveaux : variable, expression et instruction, un aperçu du traitement que nécessite cette construction. La liste obtenue sera déjà très proche des formes types et dans le cas particulier d'une instruction CASSANDRE simple, non conditionnée par une impulsion et ne comportant aucune expression conditionnelle, ce sera la forme type elle-même.

a) Traitement des variables

Le traitement d'une variable CASSANDRE provoquera, parallèlement à son analyse, la construction d'une sous-liste ayant la structure décrite comme forme type et ne se différenciant de celle-ci que par le fait que les expressions arithmétiques et logiques qui constituent éventuellement les indices ou les bornes peuvent être conditionnelles.

Le traitement des variables généralisées vise essentiellement à ramener leur écriture à une forme unique. Une variable généralisée étant une suite d'opérateurs portant sur une "variable élémentaire", c'est à dire soit une variable CASSANDRE, soit une expression, les diverses écritures possibles sont conséquences des propriétés propres ou réciproques de certains des opérateurs et de la distributivité de quelques uns d'entre eux.

Les principaux traitement effectués sur une variable généralisée sont donc les suivants :

. suppression de parenthèses redondantes avec les priorités des opérateurs

. addition du couple (1, 2) quand le couple d'entiers est omis après la transposition

. suppression d'opérateurs sans effet après l'opération réduction

. développement quand il y a propriété de distributivité
(la variable généralisée peut alors éventuellement devenir une expression vis à vis du niveau facteur immédiatement supérieur).

. inversion de l'opérateur "-" et de l'opérateur qui lui succède (inversion possible avec la transposition ou la rotation) de façon à rapprocher l'opérateur négation de la variable élémentaire

. lorsque le dernier opérateur est la négation et que la variable élémentaire est une expression, on la remplacera par l'expression obtenue par application des règles de Demorgan et de double négation et l'opérateur "-" sera supprimé de la liste des opérateurs

. suppression des groupements "--" et "∿ N1 N2 ∿ N1 N2"

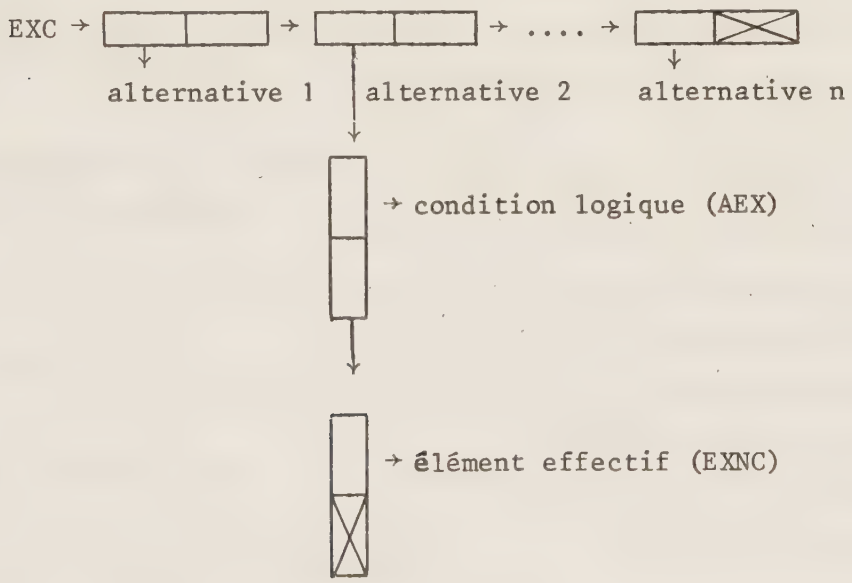
. remplacement des groupements "D|N1|D|N2|" et "R|N1|R|N2|" respectivement par D|N1+N2| si N1 . N2 > 0 et par R|N1+N2| si N1 + N2 ≠ 0 ou suppression si N1 + N2 = 0.

b) Traitement des expressions

Après traitement, les expressions pourront se présenter sous l'une ou l'autre des formes suivantes : sous la forme dite "polynomiale" retenue comme forme type des expressions ou sous la forme dite "conditionnelle".

La forme conditionnelle sera représentée par une liste d'alternatives, chaque alternative étant constituée par une condition et par un élément effectif. Ce dernier sera lui-même une expression sous forme polynômiale.

En effet, toute condition trouvée à un niveau quelconque dans une expression sera "remontée" au premier niveau. De ce fait seront considérées comme conditionnelles les expressions de genre CASSANDRE conditionnel, ainsi que toute expression où apparaît un terme, un facteur ou un élément de variable généralisée conditionnel.



Liste représentant une expression conditionnelle

Notons que dans la forme conditionnelle CASSANDRE d'une expression le "sinon" est obligatoire ; les alternatives se présenteront donc toujours en nombre pair, puisqu'il en sera créé une correspondant à la partie "alors" et une autre correspondant à la partie "sinon".

Les formes conditionnelles seront, à leur tour, traitées de façon particulière lors de la phase finale de la construction des formes types afin de faire "remonter" les conditions jusqu'au niveau instruction. C'est ainsi que l'on ne trouve plus comme forme type que la forme polynômiale.

Le traitement des expressions peut se décomposer en deux classes distinctes de transformations. D'une part, les transformations qui visent à repérer les facteurs ou termes conditionnels qui conduiront à mettre l'expression sous forme conditionnelle et d'autre part, les transformations qui ont pour but de ramener à deux couches les expressions non conditionnelles ainsi que les éléments effectifs des différentes alternatives.

Les traitements concernant les éléments (termes ou facteurs) conditionnels consisteront essentiellement à :

. si l'élément de niveau supérieur jusque-là analysé (expression ou terme) n'était pas conditionnel, le transformer en élément conditionnel

. sinon imbriquer les conditions et mettre à jour l'élément effectif.

La mise sous forme de sommes de produits des expressions non conditionnelles et des éléments fera appel aux transformations suivantes :

. développement des produits de facteurs, lorsque l'un de ces derniers est en fait une expression non conditionnelle

. remplacement des fonctions booléennes par leur expression associée dans la base et, ou, négation

. application des règles de simplification des monômes et polynômes arithmétiques et booléens.

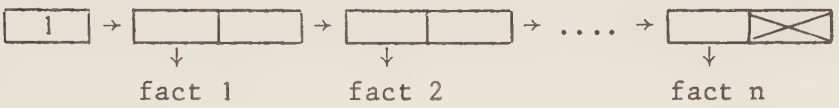
α - Algorithme de formation de la structure d'une expression

Chaque fois que l'on entre, au cours de l'analyse syntaxique, dans une expression, ou dans l'un de ses éléments : terme, facteur et éventuellement concaténé, on initialise un certain nombre d'éléments de listes ou de sous-listes, dont les pointeurs sont conservés dans une pile de récursivité.

L'algorithme utilisé peut, par exemple, être expliqué de façon schématique sur le traitement permettant de construire un monôme (ou terme) et de le ranger dans une expression.

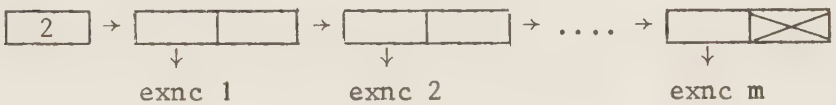
Suivant sa nature, chaque facteur analysé aura été traité et accroché à une liste différente.

1. Liste des facteurs simples

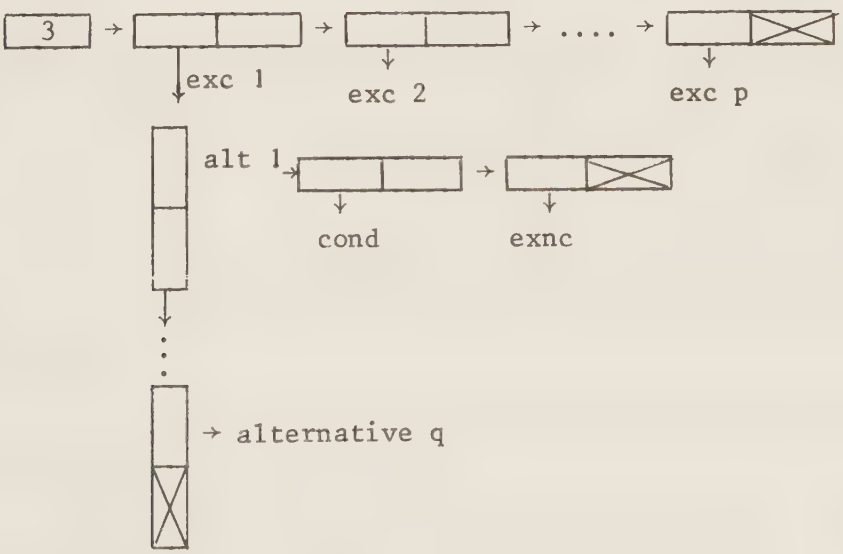


Chaque nouveau facteur simple sera simplement ajouté à la liste [1] pré-existante.

2. Liste des facteurs expressions non conditionnelles



3. Liste des facteurs expressions conditionnelles



Lorsqu'un facteur sera reconnu "expression" par l'analyse il se présentera, après traitement, soit sous la forme polynômiale, soit sous la forme conditionnelle, et sera, selon le cas, accroché à la liste [2] ou à la liste [3] préexistante.

A la fin de l'analyse d'un monôme au moins l'une des trois listes contiendra au moins un élément et dans la pile de récursivité nous trouverons successivement, en partant du sommet, les pointeurs 1, 2, 3, puis I, II et III correspondant aux différents types de termes éventuellement déjà analysés (simples, expressions non conditionnelles, expressions conditionnelles).

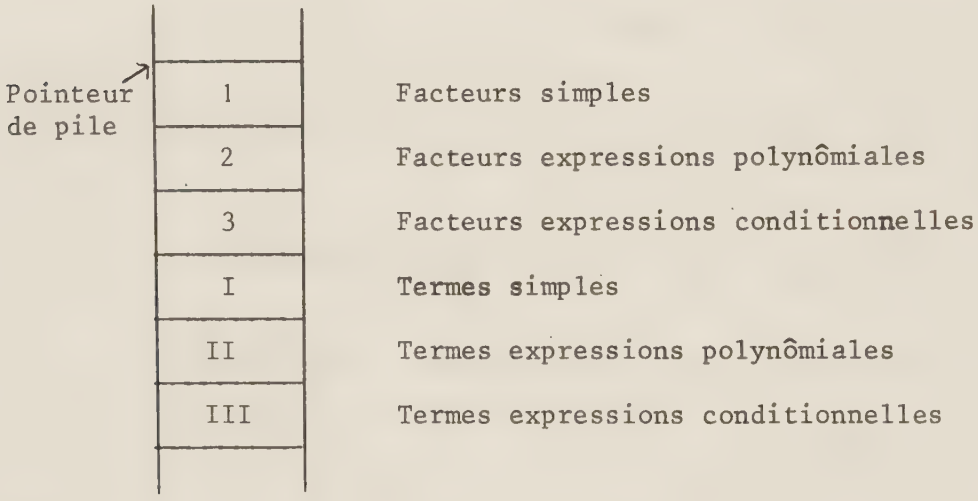
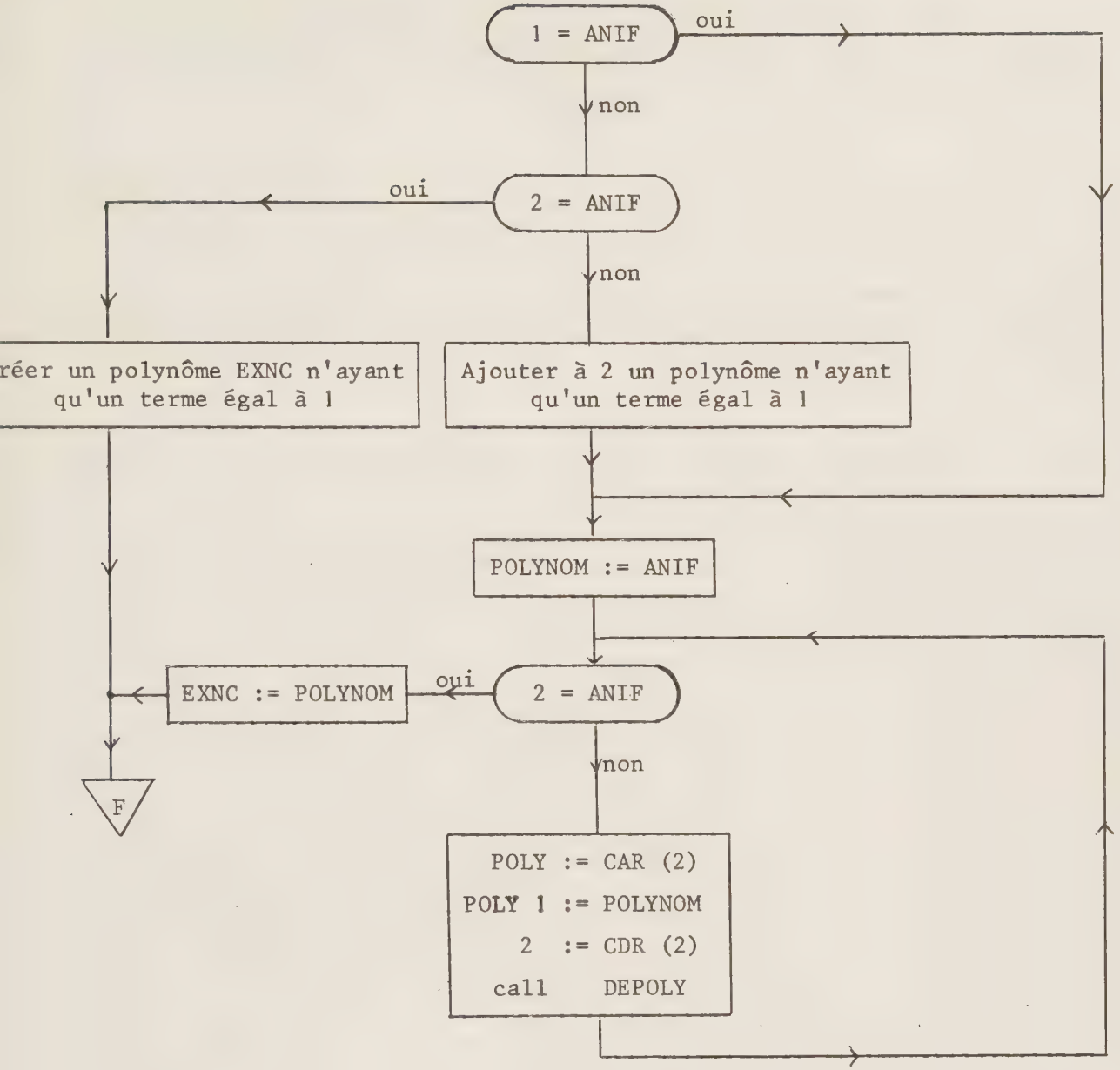


Image de la pile de récursivité à la fin de l'analyse d'un monôme

On dispose de quatre fonctions sémantiques essentielles PLACE, MIXTE IMBRIQ et TRITUR qui sont chargées de construire le monôme à partir de ses éléments rangés en 1, 2 ou 3 et de le ranger à leur tour en I, II ou III suivant sa nature en tant que terme.

La fonction TRITUR crée à partir de 1 et 2 un ou plusieurs termes.



Organigramme de TRITUR

Commentaires :

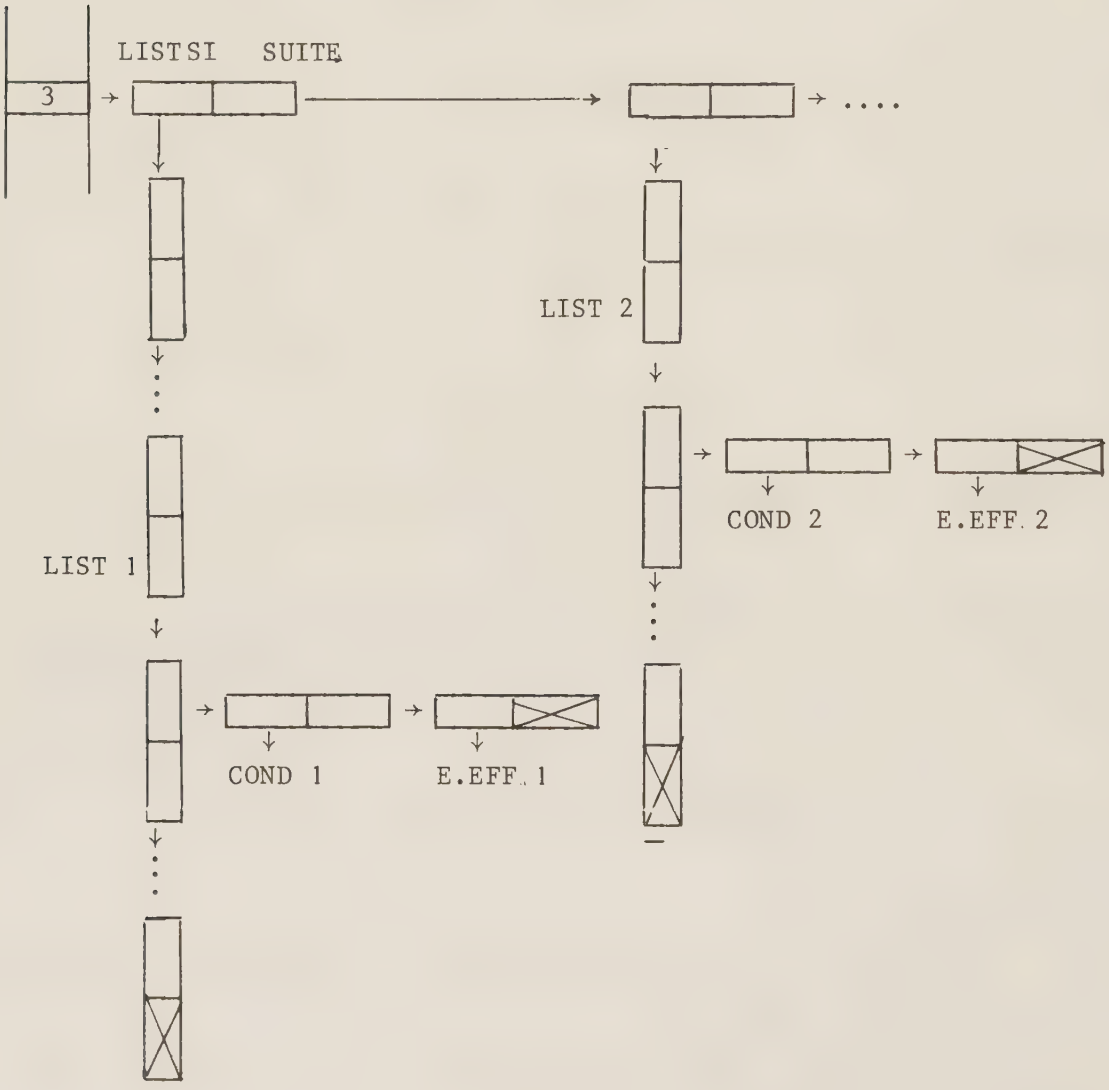
. Un pointeur de liste est initialisé par chargement du pointeur spécial de fin de liste : ANIF

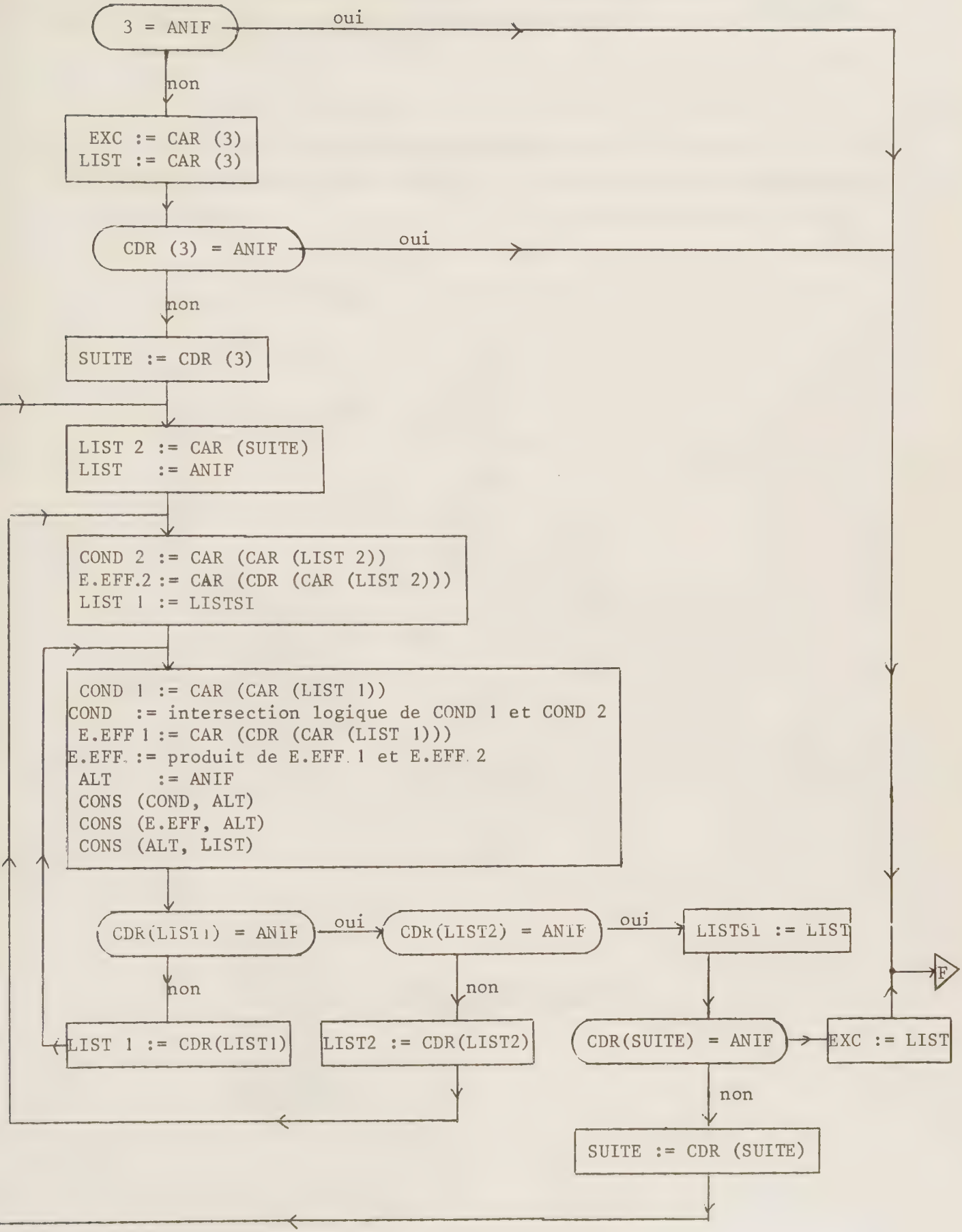
. Les fonctions CAR et CDR permettent d'obtenir respectivement la partie gauche ou droite d'un élément de liste dont on connaît le pointeur.

. La fonction DEPOLY effectue formellement le produit de deux formes polynômiales pointées respectivement par POLY et POLY 1 ; le résultat est une forme polynômiale pointée par POLYNOM.

IMBRIQ et MIXTE : ces fonctions ne sont utilisées que dans le cas où 3 n'est pas vide. Alors, IMBRIQ imbrique les conditions et effectue le produit des éléments effectifs correspondants. Le résultat est un ensemble d'alternatives dont MIXTE remplace tous les éléments effectifs par leur produit avec le résultat de TRITUR.

La liste sur laquelle travaille IMBRIQ est explicitée ci-dessous :

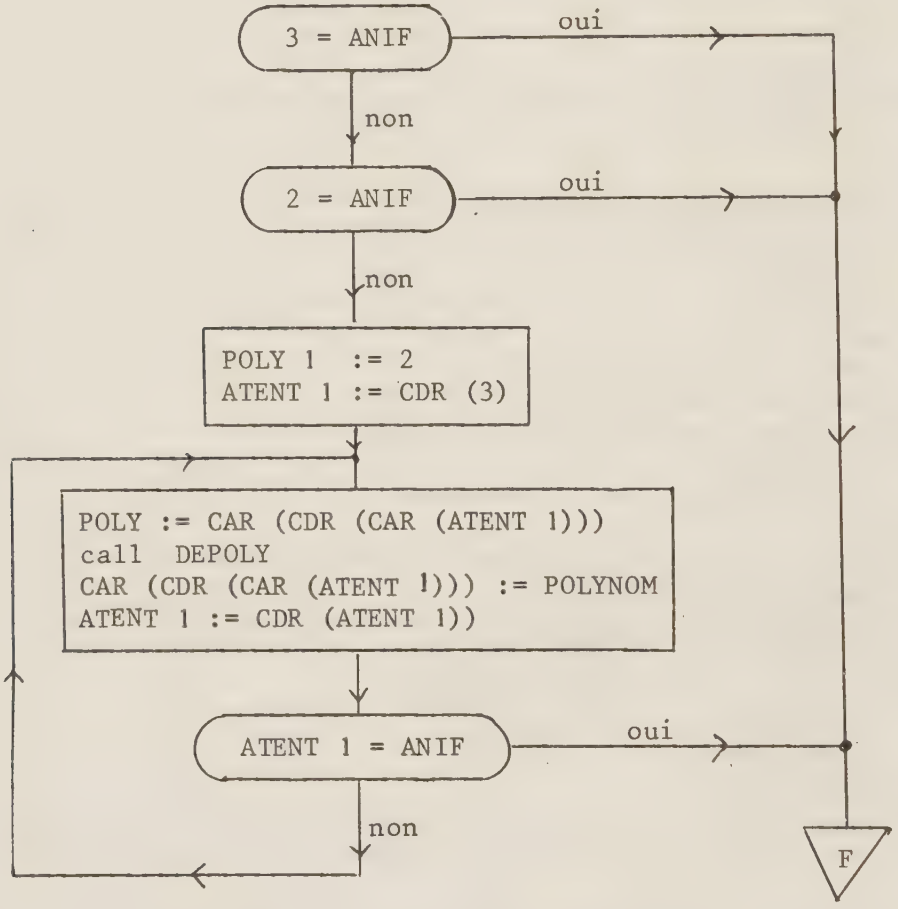




Organigramme de IMBRIQ

Commentaire : La fonction CONS (A, B) permet d'ajouter à la liste dont le pointeur est donné en deuxième paramètre un élément ayant le premier paramètre comme partie gauche.

La fonction MIXTE travaille sur les formes résultant du traitement de TRITUR et de IMBRIQ et qui sont respectivement une forme polynômiale EXNC et une forme conditionnelle EXC.



Organigramme de MIXTE

La fonction PLACE range le résultat des trois fonctions précédentes, c'est à dire un ou plusieurs termes conditionnels ou non, dans le type de terme correspondant à sa nature.

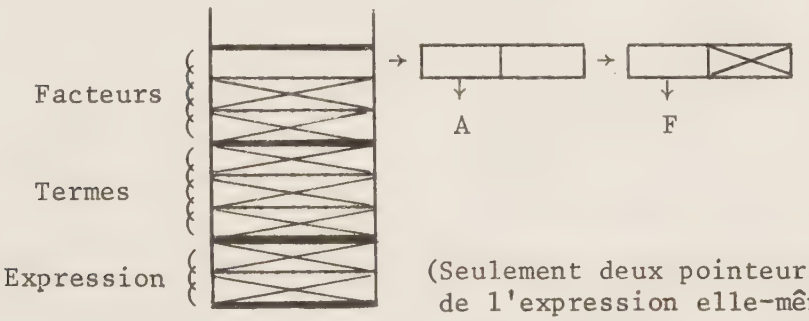
Ces quatre mêmes fonctions sémantiques, appliquées à des traitements différents des éléments effectifs, permettent de ranger un terme dans une expression arithmétique ou booléenne, un élément de concaténé dans une variable généralisée...

β - Exemple

Les schémas ci-dessous donnent l'état des pointeurs de la pile de récursivité au cours de l'analyse de l'expression logique suivante :

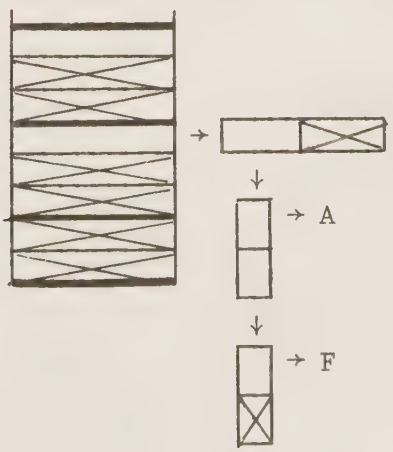
A . F + B . (C + D . G) . E . (si C1 alors X sinon Y + Z)

(1) Après analyse du premier terme

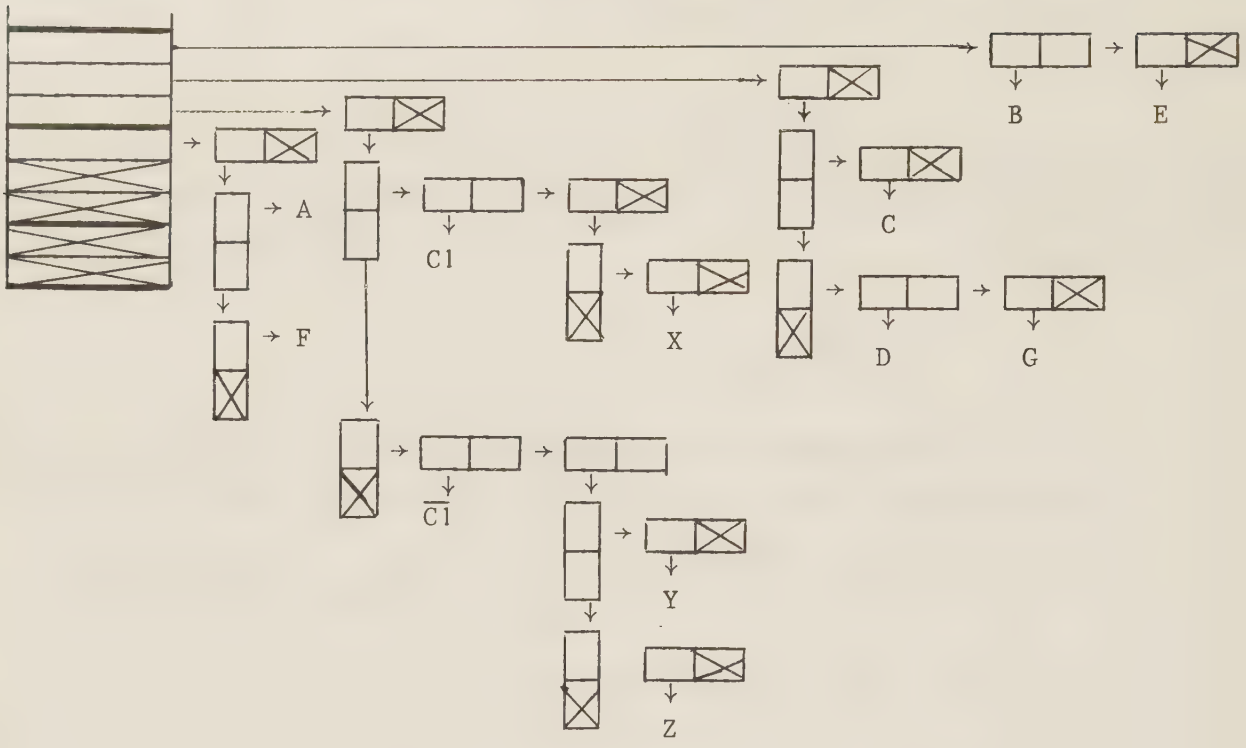


(Seulement deux pointeurs sont initialisés au niveau de l'expression elle-même, l'un correspondant à la forme polynômiale et l'autre à la forme conditionnelle)

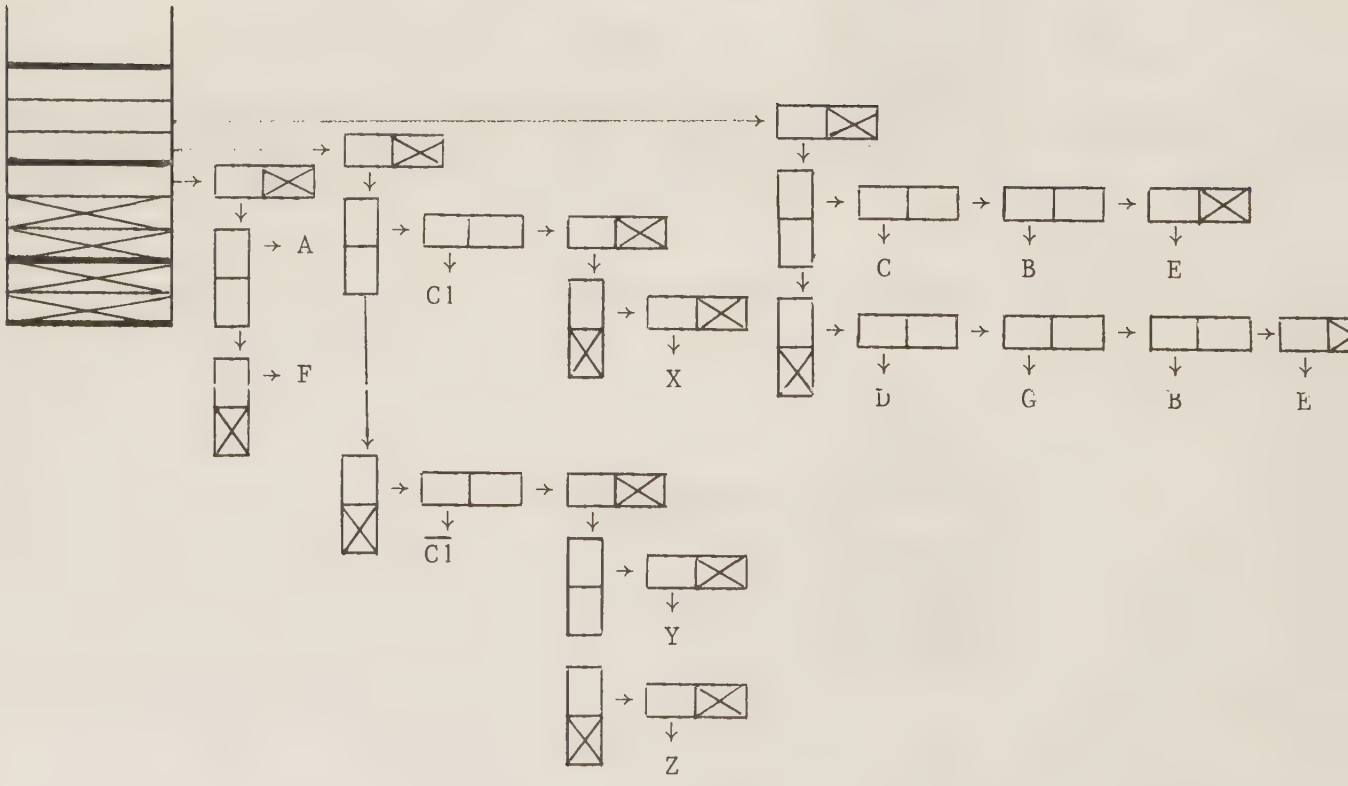
(2) Après rangement du premier terme



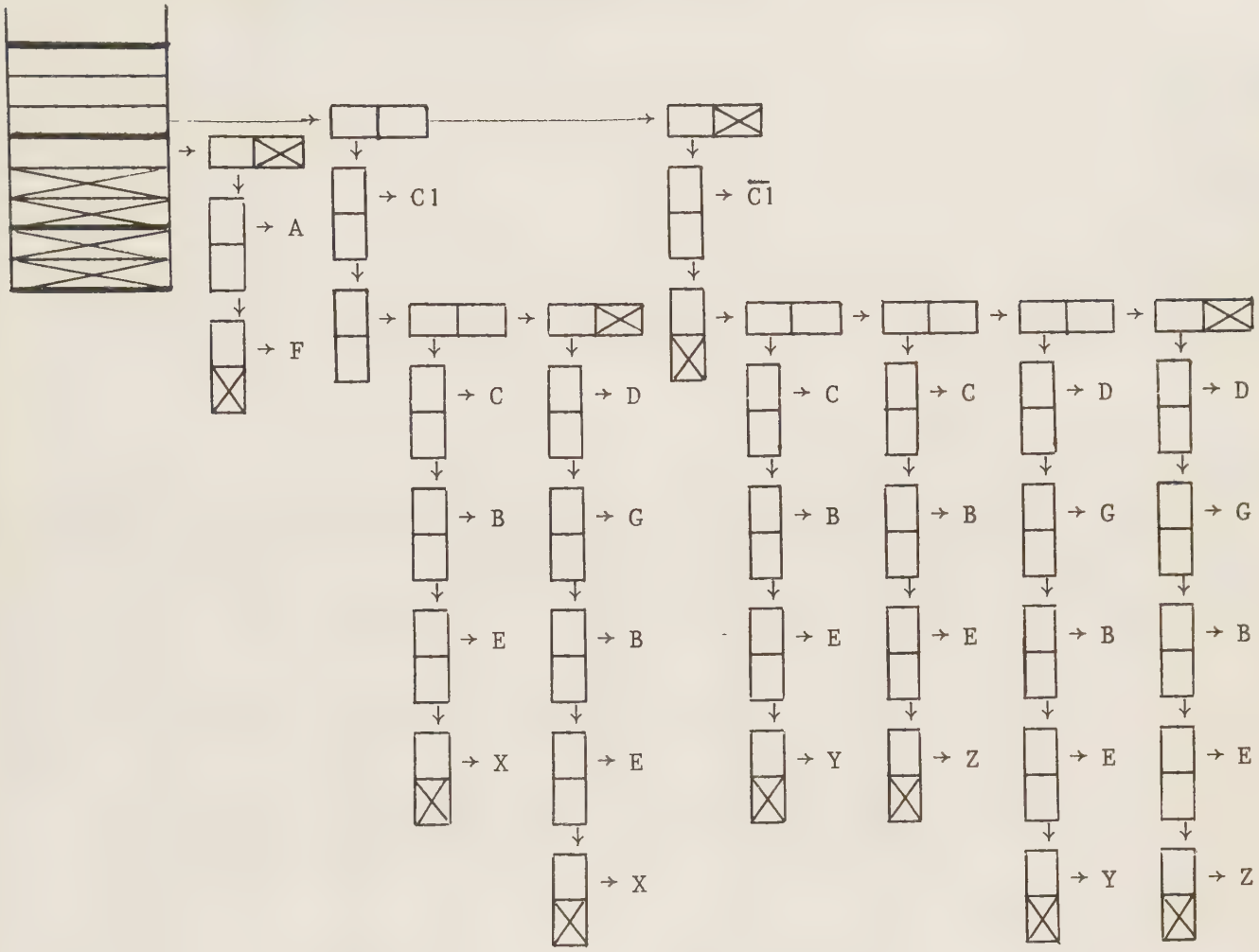
(3) Après analyse du deuxième terme



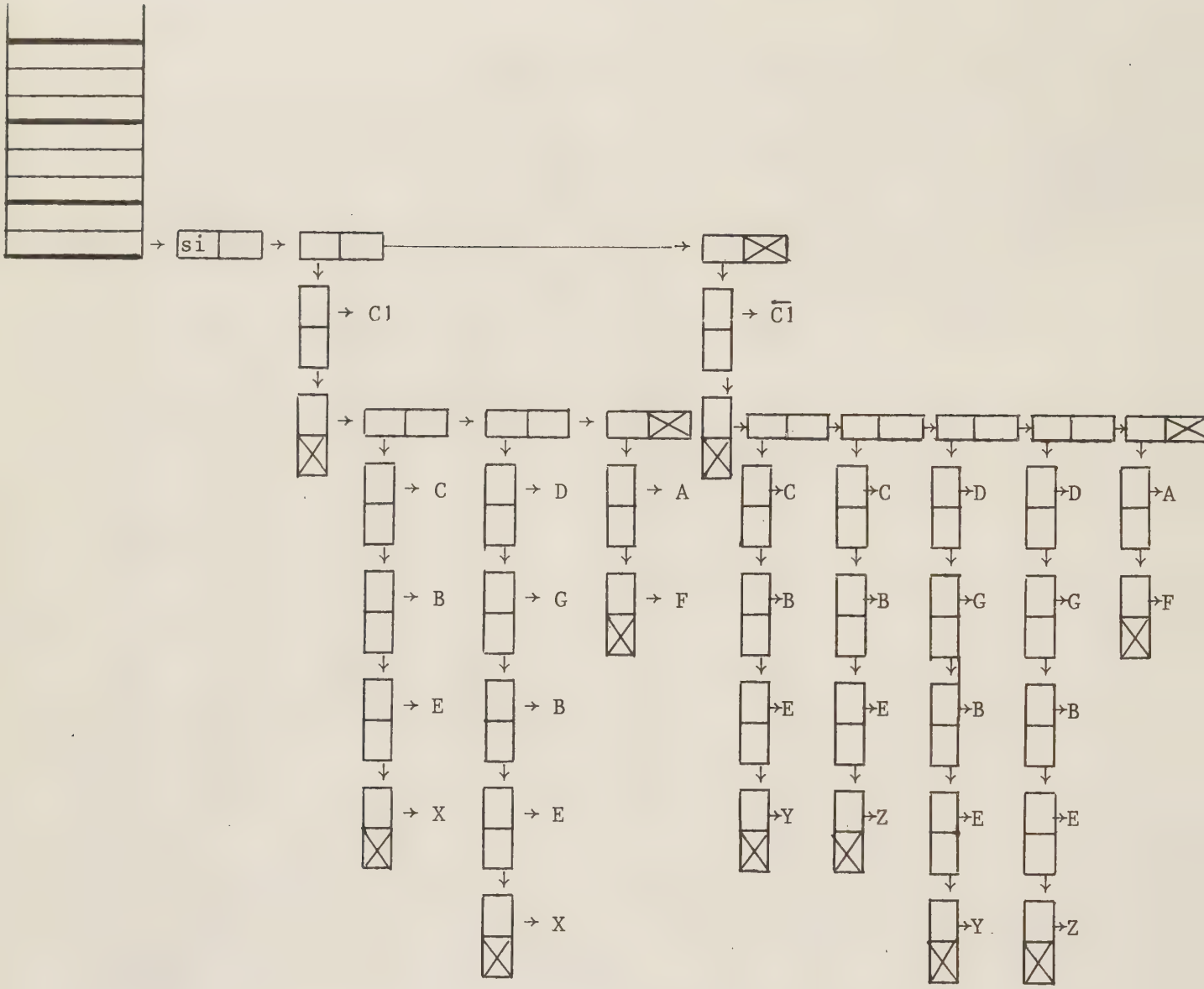
(4) Après TRITUR du deuxième terme



(5) Après IMBRIQ et MIXTE du deuxième terme



(6) Après analyse et traitement complet de l'expression



c) Traitement des instructions

Au cours de l'analyse, le traitement des instructions vise à construire pour chaque instruction CASSANDRE de l'unité décrite, une liste correspondante.

Pour les instructions simples, la liste obtenue est très proche de la forme type. Elle en diffère essentiellement par le fait que les expressions y figurant, à quel que niveau que ce soit, peuvent aussi bien être de type polynômial que de type conditionnel.

Le principal traitement sur les instructions simples consiste à remplacer les instructions du genre connexion de signal, chargement de registre ou génération d'impulsion par l'instruction du genre connexion d'unité équivalente. Nous avons vu que cette équivalence était possible lorsque la partie droite des instructions remplacées avait comme seul élément un "signal unité".

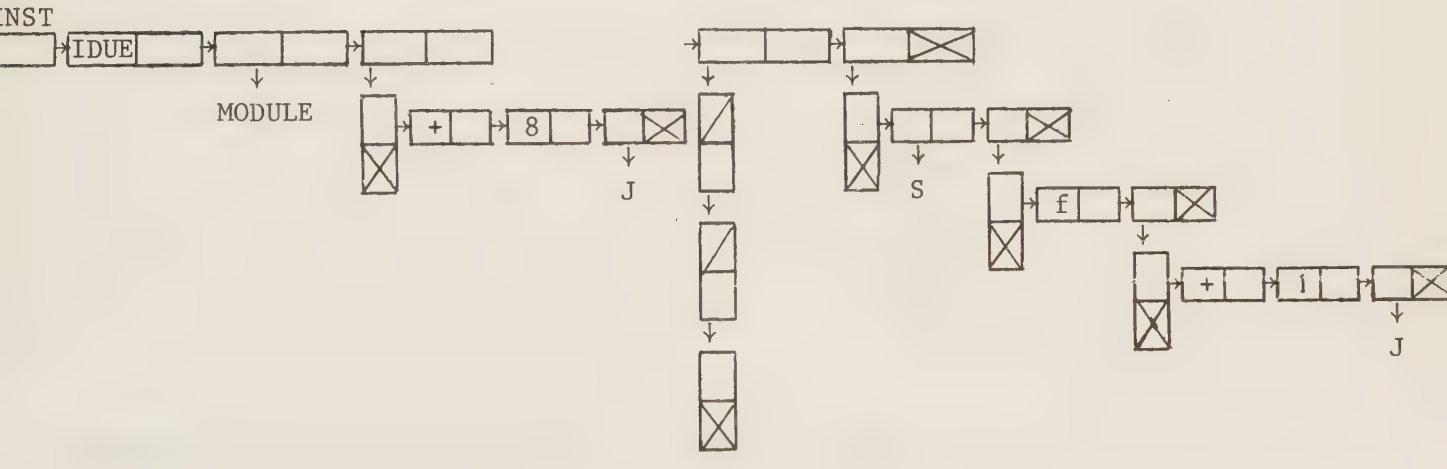
Exemple : L'instruction suivante :

S(J) = MODULE |8.J| (, , ; *) ;

aura comme liste représentative celle de l'instruction :

MODULE |8.J| (,, ; S(J)) ;

Soit :



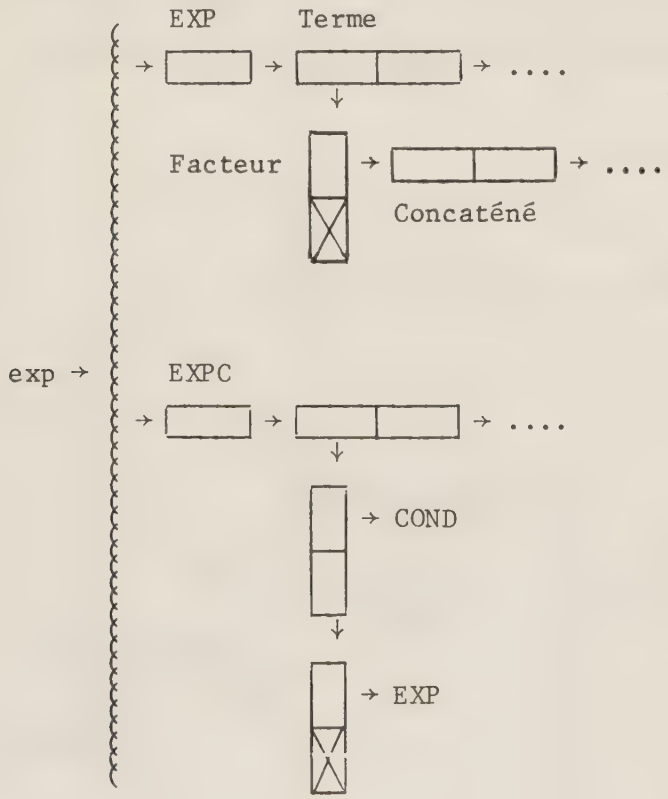
Par contre, la liste construite après analyse d'une instruction composée présente d'importantes différences avec les formes types. Outre la remarque faite précédemment sur les deux formes possibles d'expressions y figurant, il faut encore noter que leur portée n'est pas une instruction mais un ensemble

d'instructions où tous les genres peuvent figurer et que les instructions conditionnelles se présentent comme un ensemble d'alternatives.

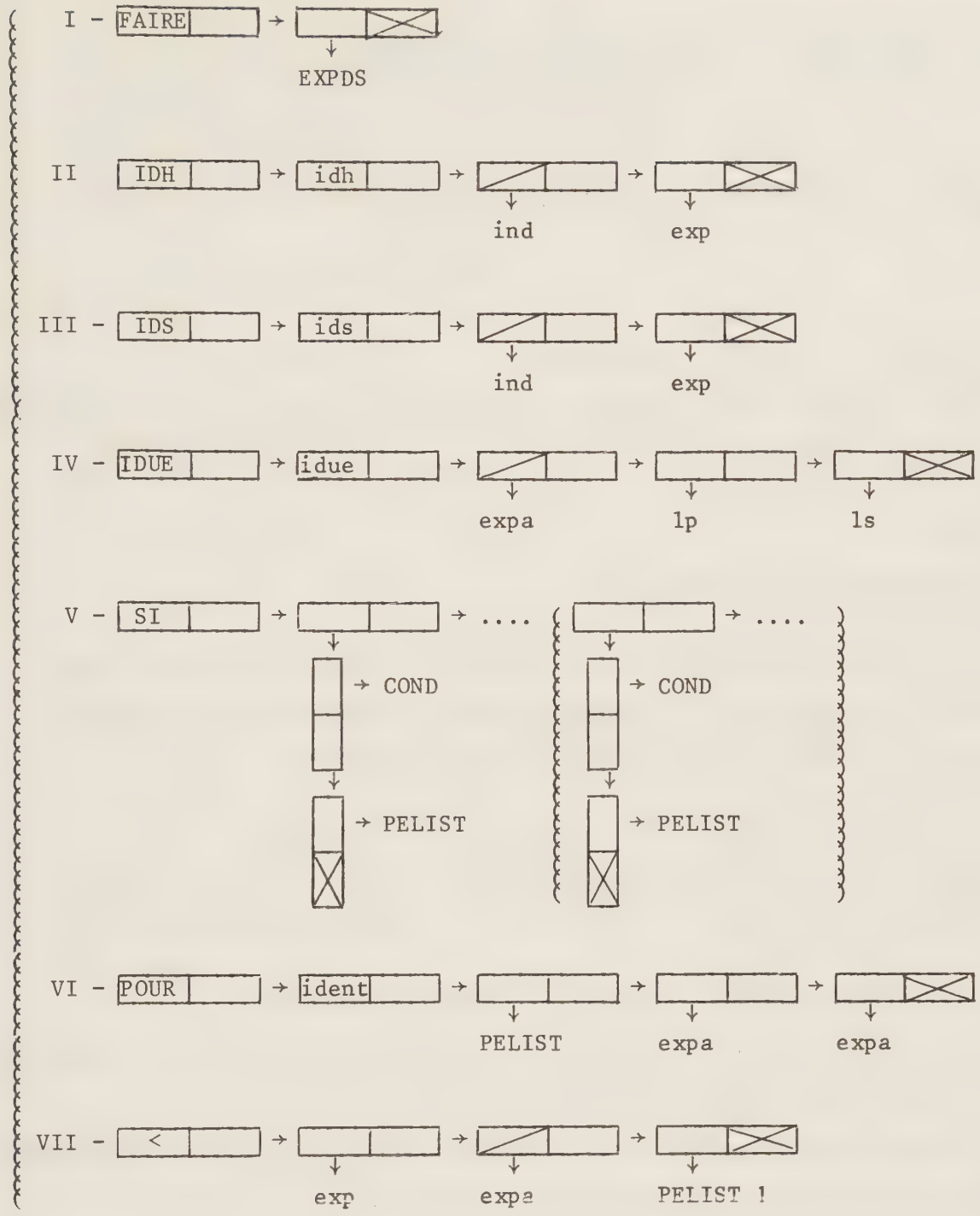
Enfin, lorsqu'une instruction (ou un ensemble d'instructions) simple ou composée est conditionnée par une impulsion, celle-ci figure explicitement dans la liste correspondante.

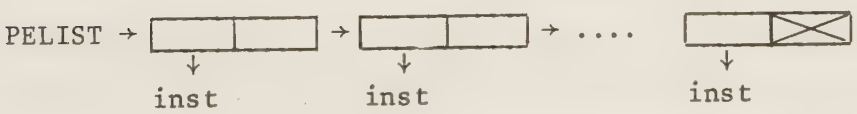
Ci-dessous est donnée une description des listes construites après analyse d'une instruction CASSANDRE. Les pointeurs désignant les expressions sont représentés par des noms écrits en lettres minuscules et recouvrent ainsi les formes polynômiale et conditionnelle.

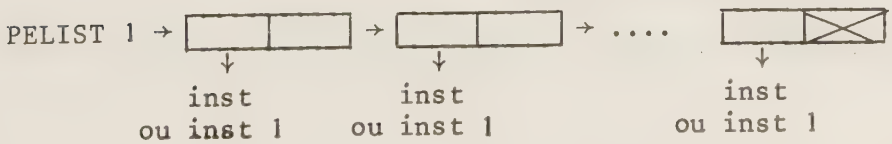
Exemple :



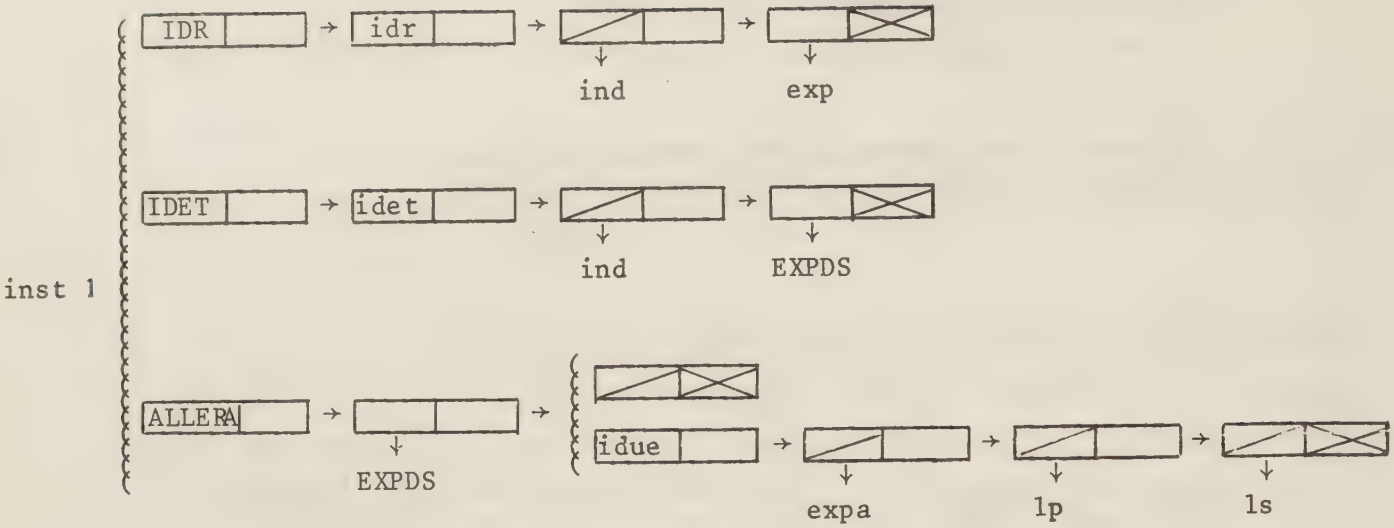
Par suite, pour avoir la liste descriptive d'un élément quelconque dont le nom est écrit en lettres minuscules, il suffira de remplacer dans la forme correspondante toute occurence d'expression par le nom de celle-ci en minuscules.







Rmq : dans les "inst" appaissant dans PELIST 1 remplacer PELIST par PELIST 1



3 - CONSTRUCTION DES FORMES TYPES

C'est à partir de la liste établie au cours de l'analyse d'une instruction CASSANDRE que la fonction sémantique ELEM construira la forme type, ou l'ensemble des formes types, lui correspondant.

Cette fonction ELEM a deux objectifs principaux :
si nécessaire, modifier le type CASSANDRE de l'instruction, et "éclater" l'instruction analysée en un certain nombre d'instructions élémentaires.

Transformations visant à la modification du type :

. une instruction composée peut devenir conditionnée par une impulsion si elle porte sur une instruction de ce genre.

. une instruction peut devenir conditionnelle si elle porte directement sur une expression ou instruction de ce genre.

Recherche des instructions élémentaires :

. une instruction "pour", "si" ou conditionnée par une impulsion sera éclatée en autant d'instructions élémentaires du même type qu'il y a d'éléments dans la portée de l'instruction analysée.

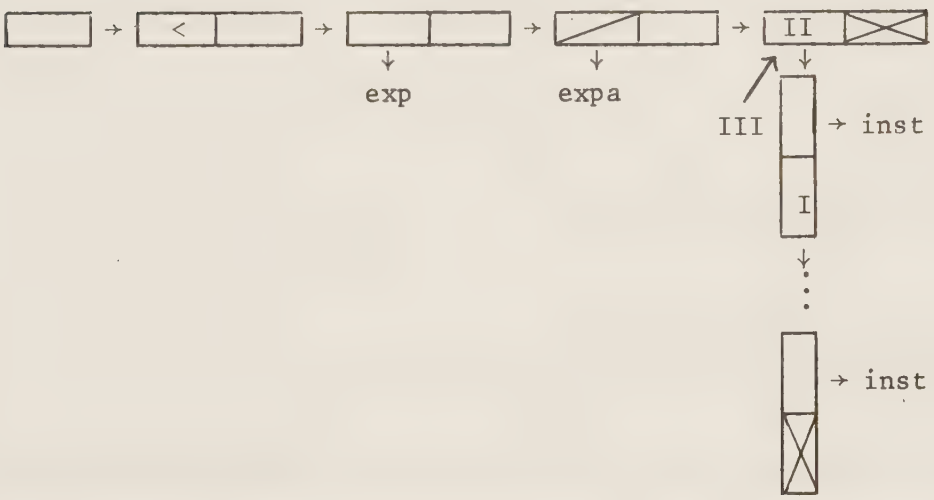
. imbrication de conditions et mise à jour des éléments effectifs.

La liste obtenue après analyse est parcourue entièrement par ELEM qui stocke au passage dans une pile spécialisée un certain nombre d'informations qui lui permettront de construire les différentes formes types élémentaires que l'on peut en tirer. Après la construction de chaque forme type on fera appel à une fonction de "codage" décrite dans le paragraphe suivant.

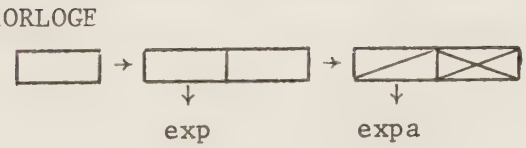
a) Stockage des informations

La présence d'une impulsion est repérée grâce au symbole de base "<". On conservera alors d'une part, dans un pointeur réservé à cet effet et nommé HORLOGE, l'ensemble constitué par l'expression de synchronisation et par le facteur de duplication éventuel et d'autre part, les renseignements qui rendront possible la répercussion de ce top d'horloge sur chacune des instructions du bloc sous sa portée.

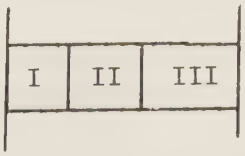
A partir de l'élément de liste suivant :



on créera :



et on remplira une ligne de la pile avec :

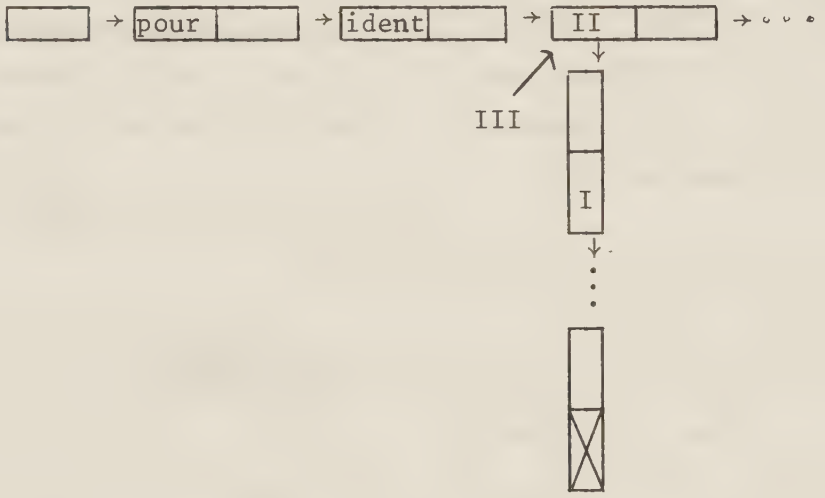


Ligne de type "BLOC"

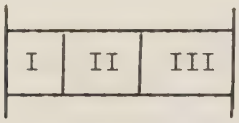
I et II sont des pointeurs alors que III est une adresse, celle de l'implantation de l'élément considéré dans l'espace liste.

D'une façon générale, chaque fois que l'on aura à conserver les renseignements nécessaires pour "éclater" une racine sur sa portée on créera dans la pile une ligne du type "BLOC".

Ainsi après repérage du symbole de base "pour" dans l'élément de liste suivant :



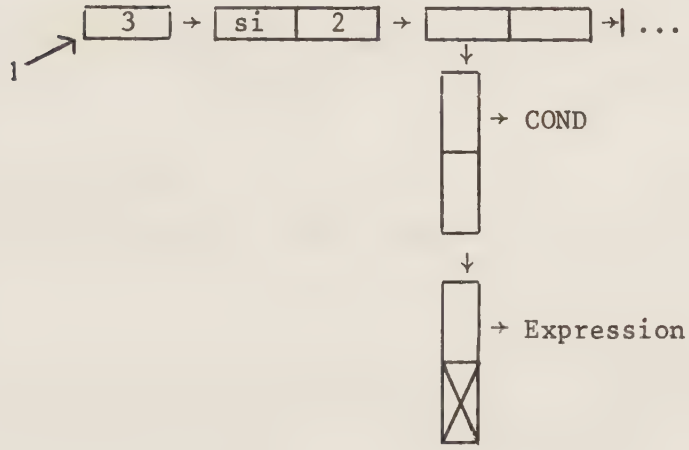
on remplira une ligne de la pile avec :



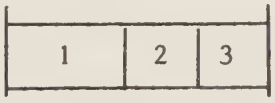
Un autre type de ligne de la pile sera créé pour garder les renseignements nécessaires aux imbrications de conditions. ELEM devra pouvoir imbriquer entre elles des instructions conditionnelles, des instructions conditionnelles avec les expressions conditionnelles y figurant éventuellement et enfin des expressions conditionnelles avec les expressions conditionnelles apparaissant à l'intérieur de celles-ci. En effet, dans le traitement des expressions nous n'avons pas considéré les deux cas particuliers d'expressions conditionnelles se trouvant en indice ou en élément d'entrée d'une unité externe utilisée comme signal unité, car, seuls les concaténés ou facteurs ou termes conditionnels nécessitent un calcul formel spécifique des éléments effectifs.

Les instructions et expressions conditionnelles se présentent l'une et l'autre comme un ensemble d'alternatives, mais, alors que les éléments effectifs des premières sont un bloc d'instructions, ceux des secondes sont eux-mêmes des expressions.

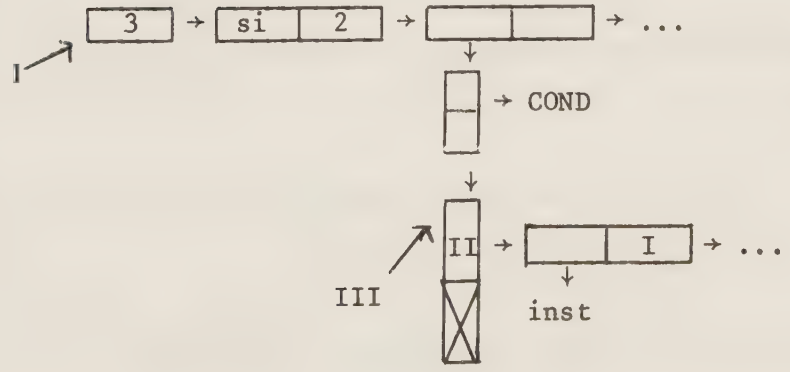
Pour les expressions conditionnelles dont nous rappelons la structure de la liste :



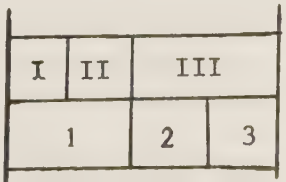
on remplira une ligne de la pile avec :



Pour les instructions conditionnelles :



on remplira deux lignes de la pile : l'une correspondant à sa nature conditionnelle et l'autre au caractère "BLOC" de son élément effectif :



b) Construction de la première forme type élémentaire

Nous avons vu comment, après un premier balayage de la liste par ELEM, ont été repérés les éléments conditionnels et les blocs d'instructions et stockés les renseignements permettant de passer à l'alternative ou à l'instruction suivante. Mais seule la première alternative ou la première instruction a été prise en compte et parcourue en détails.

Une forme type élémentaire va maintenant pouvoir être construite à partir des informations contenues dans la pile spécialisée.

Si celle-ci est vide, c'est que l'instruction CASSANDRE mise sous forme de liste au fur et à mesure de son analyse n'est autre qu'une forme élémentaire et elle se présente déjà sous forme type.

Sinon, on recherchera dans la pile toutes les informations concernant les éléments conditionnels ; on calculera l'intersection logique de toutes les conditions figurant dans les alternatives prises en compte et simultanément on remplacera chaque occurence d'élément conditionnel par son élément effectif correspondant. En outre, tout bloc d'instruction sera remplacé par l'instruction lui appartenant et retenue par ELEM.

Lorsqu'aucun élément conditionnel n'a été repéré, le traitement des blocs d'instructions suffit à fournir une liste sous forme type. Dans le cas contraire, l'instruction élémentaire est de type conditionnel ; sa condition est le résultat de l'intersection logique calculée précédemment et son élément effectif l'instruction obtenue après traitement des éléments effectifs et des blocs d'instructions.

Dans l'un et l'autre cas, la forme type élémentaire peut, en plus, être conditionnée par une impulsion décrite dans TOP.

c) Passage à la forme type élémentaire suivante éventuelle

Partant du sommet de la pile spécialisée et, selon la nature de la ligne considérée, on prendra en compte, quand elle existe, l'instruction suivante du bloc ou l'alternative suivante de l'élément conditionnel et on mettra la ligne à jour.

Lorsque le bloc (ou l'élément conditionnel) ne comprend plus d'instructions (ou d'alternatives) on cherchera à passer à la ligne précédente dans la pile. Si celle-ci n'existe pas, c'est que toutes les formes types élémentaires correspondant à l'instruction CASSANDRE ont été créées.

Au contraire, si une nouvelle instruction ou alternative existe, elle est alors analysée et peut provoquer la création de nouvelles lignes dans la pile. Après cette analyse, on est de nouveau à même de construire une nouvelle forme type élémentaire. Dans le cas où l'on est passé à une autre alternative on prendra soin de recalculer la condition correspondant à la nouvelle combinaison d'alternatives ainsi considérées.

4 - CODAGE DES FORMES TYPES

Cette phase du traitement se décompose en deux parties, d'abord la comparaison avec les formes de même type éventuellement déjà trouvées, puis, en fonction des résultats de cette comparaison, la détermination du code de l'instruction.

a) Comparaison avec les instructions types déjà codées

Par définition même des instructions toujours valides on ne cherchera à comparer entre elles que des instructions sous état et avant toute comparaison on aura déterminé le genre de la forme type à coder grâce au symbole de base CASSANDRE contenu dans le premier élément de sa liste.

Au stade de la comparaison de deux formes types on retrouve les trois différents niveaux déjà décelés : variables, expressions et instructions.

α - Comparaison entre instructions

Les formes types ont été définies et construites de telle sorte que le genre définitif en soit caractéristique. On ne comparera donc entre elles que deux instructions de même genre et on dispose de fonction de comparaison spécifique pour chaque genre.

Une fonction de comparaison parcourt simultanément les listes des deux formes à comparer et pour chaque élément doit, soit comparer directement

l'information lorsqu'il s'agit de symboles ou de pointeurs de variables, soit appeler d'autres fonctions de comparaisons particulières lorsqu'il s'agit de listes d'indices, d'expressions ou d'instructions, quand les formes types sont composées.

β - Comparaison entre expressions

Les formes types des expressions arithmétiques (EXPA) ou des expressions d'état (EXPDS), obtenues après l'ensemble des traitements formels décrits précédemment, sont telles que deux expressions ne peuvent être égales que si elles sont représentées par des formes types semblables. En particulier, elles doivent présenter à tous les niveaux de leur liste un même nombre d'éléments.

Par contre, la forme adoptée de somme de produits n'est pas caractéristique d'une expression logique (EXP) ou booléenne (EXPB ou EXPBE) et nous avons, en outre, constaté des équivalences particulières dûes à la possibilité d'utiliser des variables, dites "généralisées", par concaténation d'autres variables.

Cette dernière remarque est également valable pour les expressions de synchronisation (EXPH), puisque la syntaxe du langage CASSANDRE autorise la concaténation d'horloges.

. Première comparaison de deux expressions

Par suite du principe même des formes types, chaque fois qu'il sera fait appel à la comparaison de deux expressions on connaîtra leur genre et il ne sera jamais fait appel qu'à des comparaisons d'expressions de même genre.

Une première comparaison de deux expression consistera à comparer les polynômes qui les représentent : comparaison faite terme à terme et facteur à facteur, en tenant compte des commutativités possibles entre termes et entre facteurs et de la nature des facteurs particulière à chaque genre.

Les remarques faites en début de paragraphe permettent de se limiter à cette première comparaison pour les EXPA ou EXPDS.

. Comparaison plus approfondie

L'égalité de deux expressions logiques ou booléennes peut être vérifiée par calcul formel de leur disjonction qui doit alors être nulle. En outre, ce calcul permet de prendre en considération les équivalences dûes aux variables généralisées après un traitement particulier et supplémentaire de celles-ci.

Ces derniers calculs formels seront optionnels de façon à laisser à l'utilisateur le choix entre une comparaison plus ou moins rapide, ou plus ou moins poussée.

- Traitement des variables généralisées

Les variables généralisées sont constituées d'une ou plusieurs variables au sens large concaténées entre elles. On appellera longueur d'une telle variable le nombre de ses éléments concaténés.

1) Un premier résultat du traitement sera la détermination de l'ensemble de toutes les différentes variables au sens large apparaissant dans l'expression comprise dans sa totalité. A chacune de ses variables sera associé un entier représentant sa "longueur" ; c'est à dire l'entier 1 dans le cas d'une variable scalaire, la longueur du vecteur dans le cas d'une variable à une seule dimension et la longueur selon la première direction dans le cas d'un tableau ou d'un tenseur.

2) A partir des longueurs associées aux variables au sens large on pourra faire correspondre à chaque variable généralisée une suite croissante d'entiers appelée "découpage". Cette suite comporte autant d'éléments que la variable généralisée comporte d'éléments concaténés. Chaque entier est obtenu par addition à l'élément précédent de la longueur de la variable au sens large correspondante (le premier élément est obtenu par addition à 0).

3) A l'aide des découpages correspondant aux différentes variables généralisées figurante dans l'expression on construira le découpage "le plus fin", obtenu par union de toutes les suites.

4) Le découpage le plus fin permet alors de restructurer les variable généralisées de façon à ce qu'elles aient toutes même longueur (même nombre d'éléments concaténés). Ceci se traduira au niveau des listes représentatives de la manière suivante : on intercalera un élément de liste vide chaque fois que le découpage correspondant à la variable généralisée considérée présente un manque par rapport au découpage le plus fin.

Exemple : Reprenons l'exemple de I.2.b (p. 11).

Soient les variables :
C ; L (1 : 2) ; I, B, F, D, M (1 : 3) ; K (1 : 4) ; J, E (1 : 5) ; A (1 : 8)

E1 et E2 deux expressions construites sur ces variables :
E1 = A&B&C&D&E . I&J&F&C&D&L&M . A&F&K&E
E2 = I&J&B&C&D&L&M . A&B&K&E . I&J&F&C&D&E . A&F&C&D&L&M

L'expression E1 a comme ensemble de variables au sens large :

{A, B, C, D, E, I, J, K, L, M}

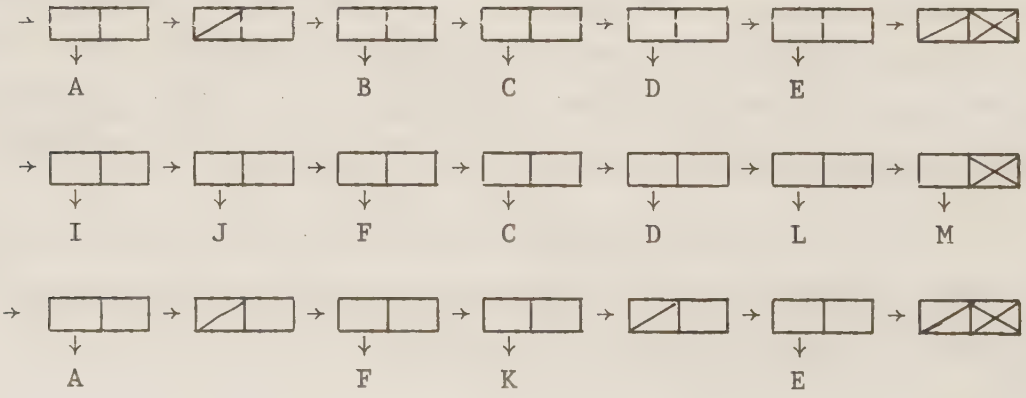
et comprend 3 variables généralisées auxquelles correspondent les découpages suivants :

(8, 11, 12, 15, 20)
(3, 8, 11, 12, 15, 17, 20)
(8, 11, 15, 20)

Son découpage le plus fin sera donc :

(3, 8, 11, 12, 15, 17, 20)

Ci-dessous sont données les listes représentatives des variables généralisées restructurées en fonction de ce découpage le plus fin :



- Comparaison des expressions

Avec cette nouvelle structure des concaténés, on peut envisager une comparaison utilisant le calcul formel de la disjonction. La comparaison se décompose en plusieurs phases successives et éliminatoires.

1) Comparaison des ensembles de variables

Les expressions à comparer ont été simplifiées au maximum au cours de leur mise sous forme de sommes de produits ; on peut donc conclure à la nécessité de la présence des variables qui restent. Deux expressions ne peuvent être égales, si leurs ensembles de variables sont différents.

2) Comparaison des découpages les plus fins

Les opérations d'union et d'intersestion sont étendues à des variables à plusieurs dimensions par calcul composante à composante. Deux expressions ne peuvent être égales que si leurs formulations développée composante à composante sont les mêmes, donc si leurs découpages les plus fins sont égaux.

3) Calcul de la disjonction

Ce calcul devra se faire progressivement et s'arrêter dès qu'un monôme évalué n'est pas nul ; les deux expressions ne pouvant être égales que si tous les monômes obtenus par développement de leur disjonction sont nuls.

Un monôme portant sur des variables simples ne peut être nul que s'il comporte une variable et cette même variable niée.

Un monôme portant sur des variables généralisées ne peut être nul que si, pour un certain numéro d'élément concaténé, on trouve parmi toutes les variables du monôme soit une variable au sens large et cette même variable niée, soit un élément vide succédant à une telle variable. (Il y en a obligatoirement un autre succédant à la variable niée).

Exemple : Les expressions E1 et E2 envisagées précédemment ont même ensemble de variables et même découpage le plus fin.

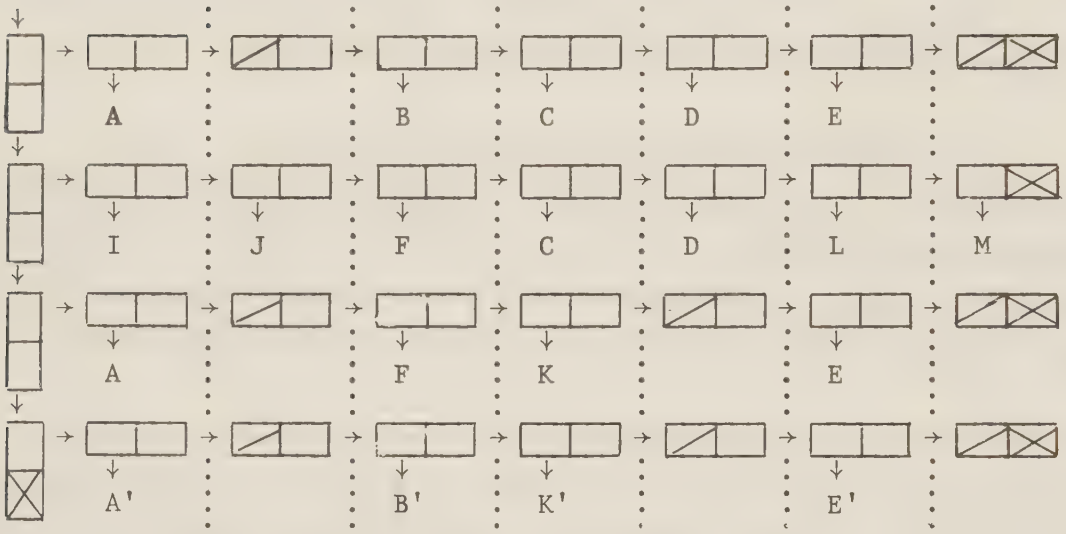
Afin d'alléger l'écriture de l'expression de leur disjonction formelle le signe "&" est omis :

E1 ⊕ E2 = (ABCDE.IJFCDLM.AFKE).(I'J'B'C'D'L'M'+A'B'K'E'+I'J'F'C'D'E'+A'F'C'D'L'
+(A'B'C'D'E'+I'J'F'C'D'L'M'+A'F'K'E').(IJBCDLM+ABKE+IJFCDE+AFCDLM)

Examinons par exemple le second monôme obtenu dans le développement de cette disjonction :

ABCDE . IJFCDLM . AFKE . A'B'K'E'

Sa représentation sous forme de liste est la suivante :



Dans chacune des colonnes déterminée par le numéro de l'élément concaténé, on trouve successivement :

A et A', un élément vide succédant à A, B et B', K et K', un élément vide succédant à K, E et E', un élément vide succédant à E.

Ce monôme des donc bien nul.

De même tous les monômes de E1 ⊕ E2 sont nuls et l'on peut conclure à l'égalité de E1 et de E2.

γ - Comparaison entre variables

Nous venons de voir comment sont prises en compte les variables généralisées, il nous reste à préciser la façon dont sont comparées les variables au sens large et enfin les variables indicées.

Le traitement formel pour la mise sous forme type a été tel que l'écriture d'une variable au sens large est unique par rapport aux différents opérateurs qui peuvent la constituer. La comparaison de deux variables au sens large conduira donc à la comparaison successive de leur liste d'opérateurs, puis de la portée de celle-ci (variables indicées, expression ou signal-unité).

Pour deux variables indicées, on comparera d'abord leur pointeur de description, puis leur liste d'indices en recherchant dans la partie descriptive de la déclaration de la variable les renseignements nécessaires lorsqu'aura été constaté l'absence d'indice, de dimension ou de borne dans l'une des deux listes comparées.

δ - Processus global de comparaison

Les formes types des instructions différentes trouvées ont été rangées dans une zone réservée à cet effet en mémoire. La structure de liste, pratique pour la construction et les comparaisons de formes types, est, par contre, trop encombrante pour être conservée telle quelle. C'est une chaîne codée parenthésée équivalente à la liste qui est stockée : les informations contenues en partie gauche des éléments de liste sont transcrites directement dans la chaîne et à chaque sous-liste correspondra une sous-chaîne entre parenthèses.

Les chaînes parenthésées équivalentes aux formes types sont données en annexe III.

La liste des instructions types est tenue à jour dans une table où les instructions de même type sont chaînée les unes aux autres.

Le processus de comparaison est alors le suivant :

On regarde dans la table si des instructions de même type que celles à coder ont déjà été trouvées. Dans la négative, l'instruction considérée est la première de ce type trouvée et il lui sera associé un code nouveau.

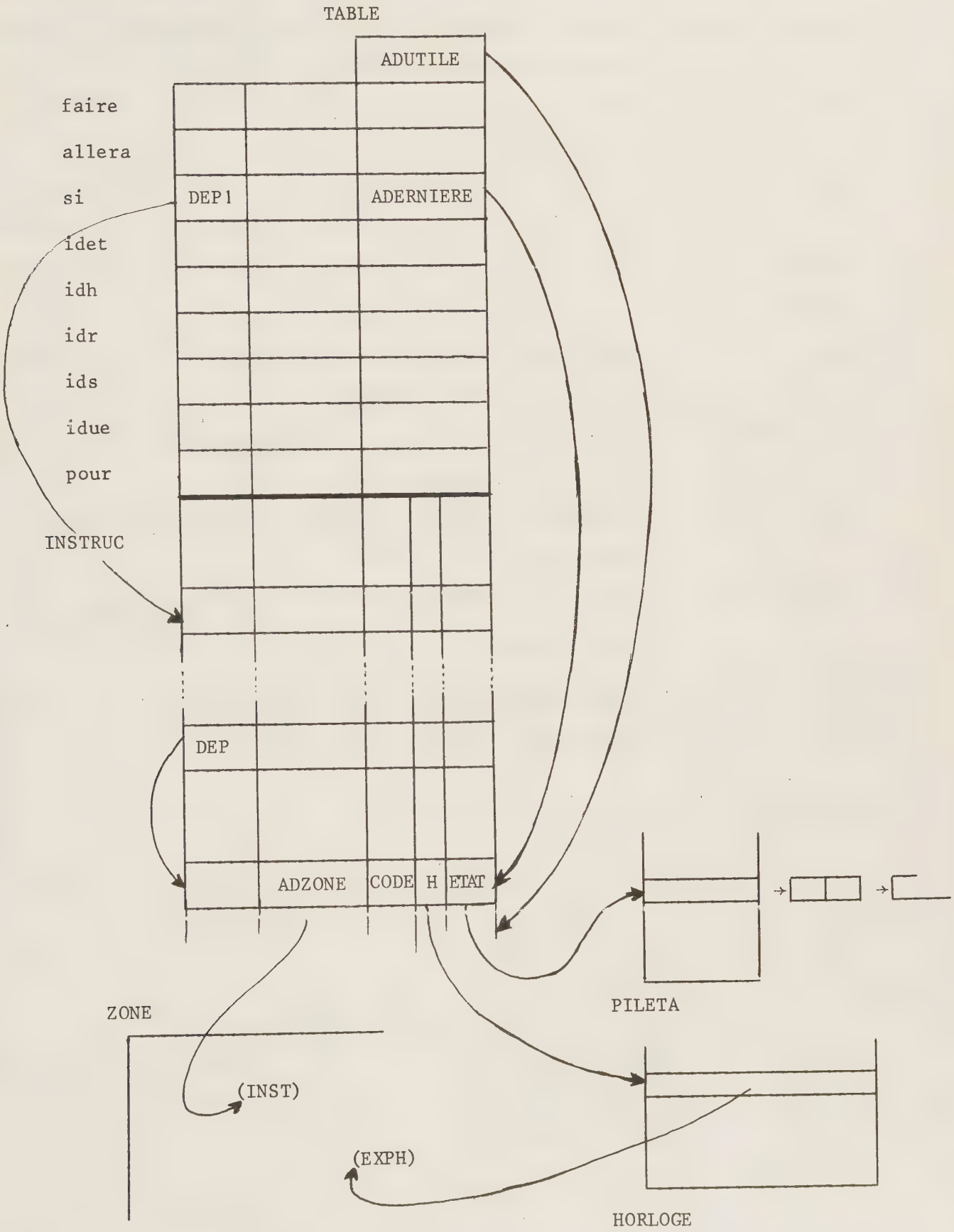
Dans le cas contraire, il va falloir comparer l'instruction à coder avec chacune des instructions de même type trouvées. Pour permettre un maximum

d'efficacité, la comparaison va porter sur les formes types associées, l'une étant sous forme de liste et l'autre sous forme de chaîne parenthésée. La comparaison des symboles sera effectuée directement entre la liste et la chaîne et au fur et à mesure des sous-chaînes rencontrées on construira les sous-listes correspondantes à partir de la chaîne.

Si l'instruction à coder se révèle équivalente à une instruction déjà trouvée, il lui sera associé le même code, sinon c'est une instruction nouvelle du type considéré et il lui sera associé un code nouveau.

Le schéma des zones de travail utilisées et l'organigramme du processus de comparaison sont donnés pages suivantes.

SCHEMAS DES TABLEAUX ET PILES UTILISEES POUR LES COMPARAISONS



Légende

TABLE et INSTRUC : Tableau contenant les informations concernant les différente instructions types retenues.

ZONE : Mémoire réservée au stockage sous forme parenthésée des instructions ou impulsions retenues.

PILETA : Pile de pointeurs de listes dont les éléments sont les différents états sous lesquels on a recontré l'instruction.

HORLOGE : Pile d'adresses dans ZONE des différentes impulsions retenue

ADUTILE : Adresse de la première ligne non utilisée de INSTRUC.

DEP1 : Déplacement en nombre de lignes jusqu'à la première instruction du type considéré.

ADERNIERE : Adresse dans INSTRUC de la dernière ligne remplie par une instruction du type considéré.

DEP : Déplacement en nombre de lignes jusqu'à la prochaine instruction du type considéré.

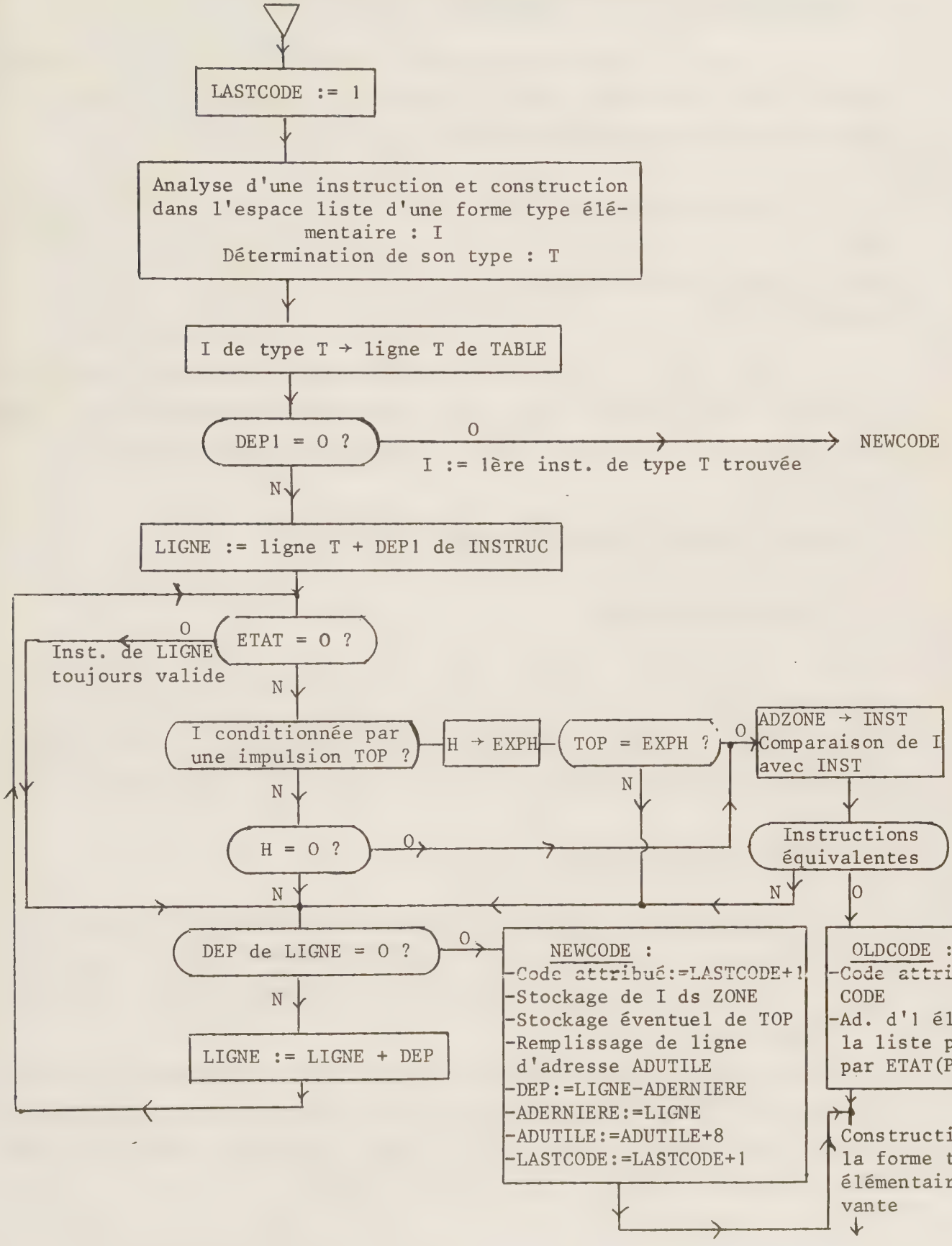
ADZONE : Adresse dans ZONE de la forme type parenthésée.

CODE : Code associé à l'instruction.

H : Déplacement dans la pile des horloges.

ETAT : Déplacement dans la pile des états.

ORGANIGRAMME DU PROCESSUS DE COMPARAISON



b) Code associé à une instruction

Le codage consiste à associer une micromot à chaque état de l'unité décrite. Chaque instruction type d'un état sera caractérisée par la mise à "1" d'une position binaire différente dans le micromot correspondant.

Une instruction déjà trouvée dans un autre état sera codée dans la même position binaire. Une instruction nouvelle sera codée dans la première position binaire trouvée en partant de la droite et non utilisée par les états déjà codés.

Les informations sur les différents codes attribués pour chaque état sont tenues à jour dans une table qui est consultée et complétée lors du codage des formes types correspondant à chaque nouvelle instruction examinée.

5 - FONCTION CONTROLE ET FONCTION OPERATIVE D'UNE UNITE

a) Fonction contrôle

L'ensemble des micromots ainsi obtenus après analyse totale de l'unité CASSANDRE considérée, constituera une première image de sa mémoire morte. On pourra alors chercher à en minimiser le coût de codage par exemple selon la méthode mise au point par Messieurs CHATELIN et HANCZAKOWSKI |7|.

b) Fonctions opérative

L'ensemble des différentes instructions types trouvées au cours de l'analyse de l'unité représente sa fonction opérative. Celle-ci sera donnée sous forme d'une unité CASSANDRE ayant les caractéristiques suivantes :

- Entête et déclarations

Ce seront celles de l'unité analysée à laquelle aura été simplement ajoutée une entrée de type signal permettant de piloter les différentes instructions types. Ce vecteur signal : &MICROCOM aura comme dimension le nombre des différentes instructions types rencontrées dans l'unité.

- Instructions toujours valides

Ce sera l'ensemble des instructions types tirées des instructions toujours valides de l'unité.

- Autres instructions

Ce sera la liste des différentes instructions types tirées des instructions sous-état de l'unité. Chacune se présentera comme une instruction CASSANDRE conditionnée par la composante de &MICROCOM correspondant à son numéro de code.

Une fonction particulière est chargée de générer cette unité à partir des informations contenues dans les tables et des chaînes codées parenthésées des instructions types.

1 - GENERALITES

Les programmes créés sont utilisés à la suite d'un système de programmes de traitements généraux conçus par Monsieur ANCEAU et Monsieur DOUSSY et réalisant les fonctions d'édition et de vérification.

a) Editeur

Ce programme lit le texte Source et fournit une chaîne dans laquelle les différents symboles de base sont codés et les blancs supprimés. Il se compose uniquement d'un module syntaxique qui provoque la sortie des caractères du code sur un fichier récupérable par le programme suivant.

b) Programme de vérification

Ce programme se compose d'un superviseur destiné à gérer les deux modules suivants.

α - Module de vérification syntaxique et d'analyse des déclarations

Ce module lit le texte édité et fournit à la bibliothèque, d'une part une chaîne complètement codée sans déclarations, d'autre part les déclarations mises sous formes de listes, et génère un texte interprétable par le module de vérification dimensionnelle. Dans l'analyse du corps du texte, des fonctions sémantiques exploitent les listes construites au moment des déclarations, pour fournir la nature et les dimensions des identificateurs rencontrés et permettent ainsi leur codage, qui est simplement l'adresse de la liste les décrivant.

β - Module de vérification dimensionnelle

Ce module est un interpréteur qui utilise le code généré à son intention par le module précédent. Ses seules sorties sont des messages d'erreurs.

Les instructions de ce code peuvent être de nature :

- arithmétique
- branchement, dans le cas de boucles "pour" ou de "si" arithmétiques

- dimensionnelle

  . prise en compte des dimensions d'un tenseur
  . modification des dimensions par les opérateurs de réduction, de concaténation et de transposition
  . tests de compatibilité de dimensions.

Ce module ajoute aux listes construites à partir des déclaractions divers renseignements issus de l'exécution des ordres arithmétiques : les numéros d'identification des unités externes utilisées dans une description.

c - Format d'une unité CASSANDRE en bibliothèque

Après passage d'un texte CASSANDRE dans la chaîne de traitements généraux on obtient un ensemble de fichiers rangés dans la bibliothèque des unités CASSANDRE précompilées.

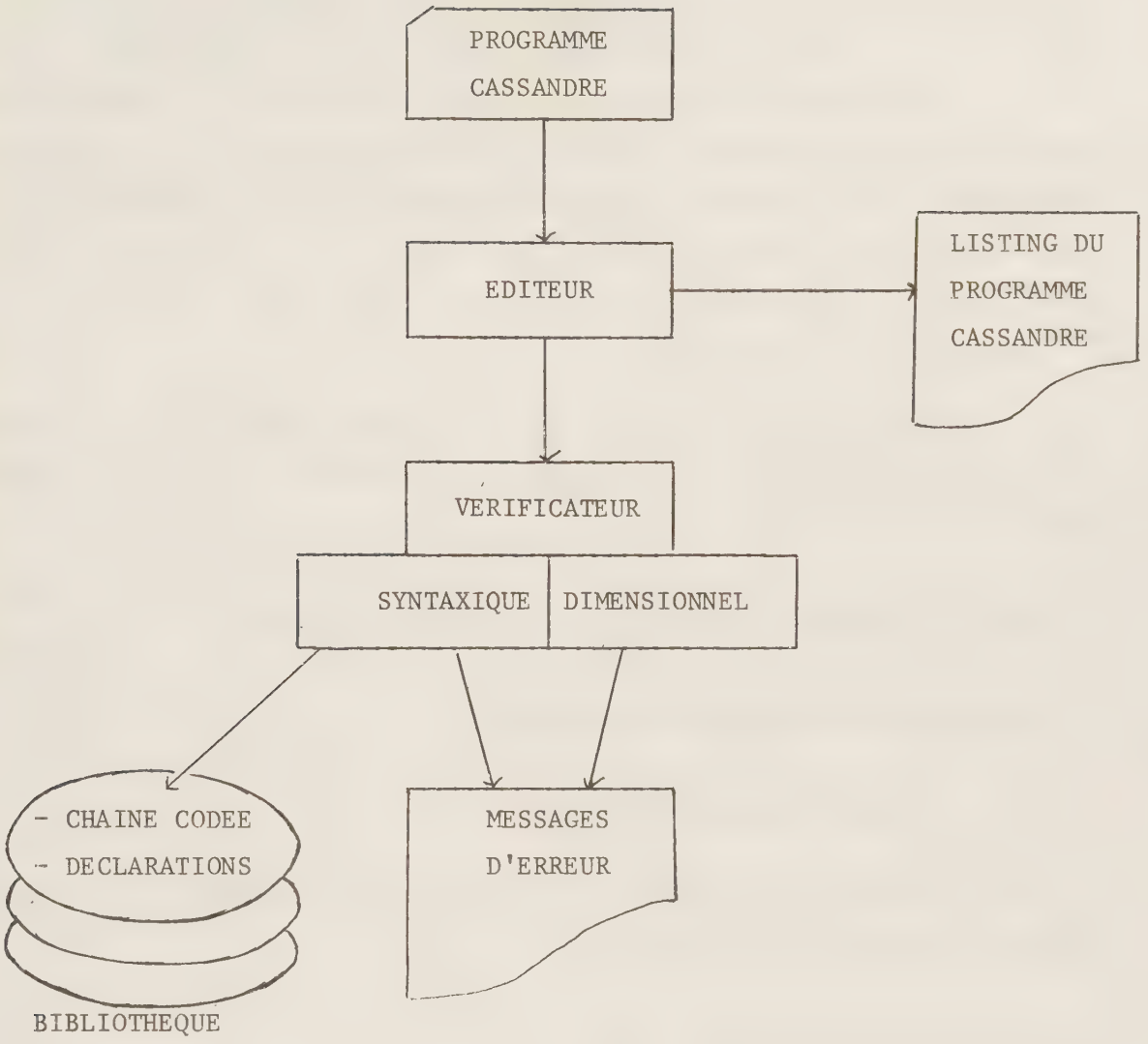
A une unité CASSANDRE correspondent deux fichiers :

- un espace liste, où sont chaînées toutes les déclarations de l'unité
- une chaîne codée, qui n'est autre que la partie fonctionnelle de l'unité, dont tous les symboles de base sont codés.

Tous les éléments déclarés (signaux, registres, etc...) sont aussi codés par leur pointeur dans l'espace liste. Aucune modification n'est apportée au texte source.

SYSTEMES DE PROGRAMMES

SECONDE PARTIE



PROGRAMMES DE TRAITEMENTS GENERAUX

2 - GENERATION DE L'ANALYSEUR SYNTAXIQUE

L'analyseur syntaxique est obtenu en utilisant le système de programmes écrits par Monsieur GRIFFITHS (IMAG) et Monsieur PELTIER (Centre Scientifique - IBM - France), qui permet la génération automatique d'un module syntaxique à partir d'une grammaire mise sous forme appropriée et pouvant contenir des appels à des fonctions sémantiques situées dans d'autres modules du programme.

La description à analyser ayant déjà été vérifiée, il est possible de partir d'une grammaire plus étendue et plus adaptée à piloter la sémantiq souhaitée.

3 - PRINCIPE DE FONCTIONNEMENT - UTILISATION DES MODULES

a) Configuration générale du système

α - Schéma synoptique

Le schéma de la configuration générale du système de programmes de traitement est donné page suivante.

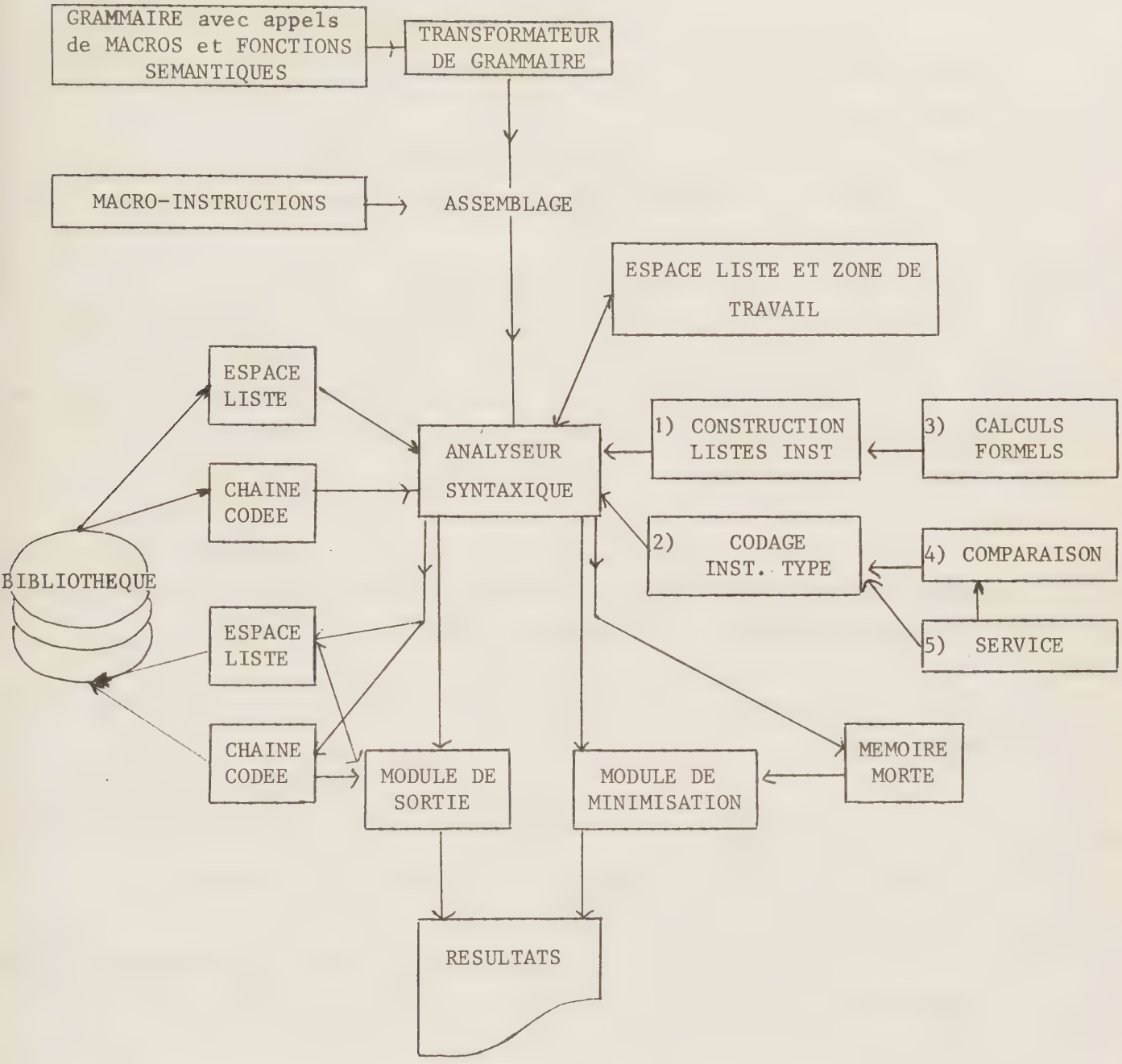
β - Description

L'analyseur reçoit en entrée la chaîne codée et l'espace liste des déclarations créées par le vérificateur à partir de la description CASSANDRE de l'unité à traiter.

La zone de travail réservée à l'analyseur se décompose en deux parties :

La première est constituée par un espace liste dans lequel il réalise la construction et les manipulations sur les formes types.

La seconde est une zone mémoire dans laquelle il stocke sous forme parenthésée les formes types des instructions différentes déjà trouvées et tient à jour des tables sur les résultats obtenus.



SCHEMA SYNOPTIQUE DE FONCTIONNEMENT

Par ailleurs, l'analyseur est en relation avec divers modules constitués chacun d'un certain nombre de fonctions qui assurent les traitements suivants :

- Module.1 : Construction des listes instructions
- Module.2 : Gestion du codage
- Module.3 : Calculs formels
- Module.4 : Comparaisons
- Module.5 : Initialisations. Créations des fichiers bibliothèque

Seules les fonctions des deux premiers modules sont appelées directement depuis l'analyseur et c'est dans ces fonctions que sont appelées celles des trois autres modules.

A la fin du traitement d'une unité, on dispose de diverses informations sur l'ensemble des instructions différentes trouvées dans la description de cette unité et le codage des états.

Les résultats se composent essentiellement de deux parties complémentaires :

(1) - Une nouvelle unité CASSANDRE comprenant toutes les instructions différentes

(2) - La liste des micromots associés à chacun des états.

Chaque micro-action du microprogramme défini par la seconde liste représentant l'occurrence d'une instruction de la première.

Ces résultats peuvent alors être respectivement pris en charge par le module de sortie écrit par Messieurs LIDDELL, GUYOT et FANTINO ou par un module de minimisation du codage des mots mémoire écrit par Messieurs CHATEL et HANCZAKOWSKI.

Tous les programmes, à l'exception de ceux du module de minimisation sont écrits en Assembleur et Macro-Assembleur. Le module de minimisation est rédigé en PL/1.

Le système est, pour l'instant, destiné à fonctionner sous CP-CMS.

b) Fonctionnement de l'ensemble

α - Analyseur

La chaîne codée relative à une instruction est analysée caractère par caractère, la progression s'effectuant sans retour arrière.

Au fur et à mesure de l'examen de cette chaîne, l'analyseur appelle diverses macro-instructions et fonctions sémantiques, ces dernières étant contenues dans le premier module, qui vont se charger de la construction progressive dans l'espace liste de la forme type correspondante. Les fonctions sémantiques du module.1 vont elles-mêmes appeler certaines fonctions du module.3 afin d'effectuer les divers calculs formels nécessaires à la construction de cette forme type.

A la fin de l'examen de la chaîne codée correspondant à une instruction CASSANDRE, la construction de la liste associée est achevée. L'analyseur appelle alors l'une des fonctions sémantiques du module.2 destinée à en tirer les formes types élémentaires et à en effectuer le codage par appel d'une fonction spécifique du type obtenu. Cette fonction va réaliser les deux phases du codage :

- L'ensemble des comparaisons avec les formes de même type éventuellement déjà trouvées est commandé par des appels à certaines fonctions du module.4.

- Le codage proprement dit de l'instruction est assuré par des fonctions appelées dans le module.2.

La construction de la mémoire morte a été effectuée au fur et à mesure de l'analyse, alors que les deux fichiers constituant la nouvelle unité CASSANDRE sont créés, grâce aux renseignements contenus dans l'espace liste et dans la zone de travail, par une fonction spéciale du module.5 écrite par Monsieur MENARD.

β - Module de sortie

Ce module permet de sortir en clair n'importe quelle unité de la bibliothèque. Il se compose de deux parties :

- Un premier programme qui ressort en clair l'espace liste, c'est à dire toutes les déclarations.

- Un analyseur qui traduit la chaîne codée en texte CASSANDRE.

γ - Module de minimisation

Les données de ce module sont, d'une part, l'ensemble des mots mémoires obtenus précédemment et, d'autre part, un certain nombre de paramètres définis par l'utilisateur, comme, par exemple, le type du codage minimal souhaité : automatique ou économique en 0 ou en 1.

Il fournit les mots mémoires recodés pour une couverture minimale adoptée et un codage déterminé.

Les exemples ci-joints ont été traités grâce à la version actuelle des programmes.

Pour chaque unité CASSANDRE on obtient successivement :

- une description en langage CASSANDRE des différentes instructions types rencontrées dans l'analyse de l'unité. En tête sont données les instructions toujours valides, puis chacune des instructions types est donnée sous forme d'une instruction CASSANDRE conditionnelle.

- un tableau de 1 et de 0 représentant la mémoire morte. Le nombre de lignes est égal au nombre d'états de l'unité considérée et le nombre de colonnes est égal au nombre d'instructions types différentes trouvées. Les "1" indiquent la présence d'une certaine instruction type sous l'état correspondant.

Le premier exemple traitant l'unité CALMAT montre clairement comment sont imbriquées instructions et expressions conditionnelles de façon à n'obtenir plus que des instructions conditionnelles à un seul niveau.

Le second exemple traitant l'unité COMPUTEI est plus instructif quant à la recherche propre des équivalences puisque de la description CASSANDRE donnée contenant 125 instructions sous état on a tiré une partie opérative ne comprenant plus que 61 instructions types.

Ces exemples montrent deux possibilités différentes de spécification de la partie opérative. Dans le premier cas, les instructions types sont conditionnées par l'état (ou l'union des états) sous lesquels elles sont validées, dans le second elles sont conditionnées par le numéro de microcommande qui peut les activer.

EXEMPLES ET RESULTATS

```
          'UNIT' MODULE ( I1 ,I2 , H ; S ) ;
'MASTERCLOCK' H ;
'REGISTER' A , P , M ;
          S := A ;
ST1 : <H> A <= A+-P ;
ST2 : <H> A <= A+P ;
ST3 : <H> M <= M.A , A <= 0 ;
ST4 : <H> P <= A ;
ST5 : <H> A <= M , M <= 1 ;
ST6 : <H> A <= I1 ;
ST7 : <H> A <= I2 ;



          'UNIT' CALMAT ( E(0:7) , H ; S (0:7)) ;
'MASTERCLOCK' H ;
'CLOCK' INCRAD ,INCRAD2 ;
 'SIGNAL' CARRY(1:7) ;
'REGISTER' ADD(1:8),RAMP(1:8),MEMORY(1:4,0:255),CODE(1:4),MATCH ;
'EXTERNAL' MODULE,ST1,ST2,ST3,ST4,ST5,ST6,ST7(,,'CLOCK';) ;
    CARRY(1:8) :=ADD(2:7).CARRY(2:7) ; CARRY(7) :=1 ;
  <INCRAD> 'IF'ADD(8)'THEN' ADD(1:7)<=ADD(1:7)+CARRY,ADD(8)<=-ADD(8);
<INCRAD2>  ADD(1:7) <= ADD(1:7)+CARRY ;
'FOR' I = 0 'TO' 8
'BEGIN'
   'FOR' J =0 'TO' 8
      'BEGIN'
       MODULE(I+8.J|(MODULE(I+8.(J+1)|(,,;*),MODULE(I+1+8.J|(,,;*),,);
      'END';
  'END';
 'FOR' I =0 'TO' 8
 'BEGIN'
   MODULE(7+8.I|(MODULE(7+8.(I+1)|(,,;*),,;);
   MODULE(I+56|(,MODULE(I+57|(,,;*),,;);
  'END';
          'FOR' I=0 'TO' 7
          'BEGIN' MODULE(I+56|(*,,;):=MODULE(I|(,,;*)  ; 'END' ;
          'FOR' J=0 'TO' 7
          'BEGIN' S(J):=MODULE(8.J|(,,;*) ;  'END' ;
 FETCH : <H> CODE <= MEMORY(,$(ADD)) , 'GOTO' DECODE , INCRAD := 1 ;
```

```
                <H> 'FOR' I=0 'TO'7
                'BEGIN' 'FOR' J=0 'TO' 7
                        'BEGIN' 'GOTO' 'IF' CODE(2) 'THEN'
                                        ('IF' CODE(3) 'THEN' ST7
                                        'ELSE' 'IF' CODE(4) 'THEN'
                                        ST6 'ELSE' ST5 )
                                        'ELSE'  'IF' CODE (3) 'THEN'
                                         ('IF' CODE(4) 'THEN' ST4
                                                'ELSE' ST3 )
                                            'ELSE' 'IF' CODE(4)
                                             'THEN' ST2
                                              'ELSE' ST1
                        'OF' MODULE |I+8.J| (,,;),'GOTO' EXECUTE ,
                'END' , 'END' ) 'ELSE'
                        ( <H>'IF' CODE(2) 'THEN'
                                ('IF' -CODE(3) 'THEN'
                                        ('IF' MATCH+-CODE(4) 'THEN'
                                        (ADD(1:4) <= MEMORY(,$(ADD))
                                          , INCRAD := 1 ,
                                        'GOTO' SECONDOF )
                                    'ELSE'(INCRAD2:=1,'GOTO' FETCH ))
                        'ELSE' 'IF' CODE(4) 'THEN' ( 'GOTO' STOP) )
                'ELSE' 'IF' -CODE(3) 'THEN' MATCH<= 'IF' CODE(4)
                  'THEN' -MATCH 'ELSE' 0;
               'IF' -/+CODE(2:3) 'THEN'('DO' FETCH));
EXECUTE : 'FOR' I=0 'TO' 7
          'BEGIN' 'FOR' J=0 'TO' 7
                  'BEGIN' MODULE|I+8.J|(,,H;)' ;  'END';
                'END' ;
        'FOR' J=0 'TO' 7 'BEGIN'
MODULE|7+8.J|(,'IF'/.(CODE+0111) 'THEN' E(J)
    'ELSE' MODULE|8.J|(,,;*),;); 'END' ;
          'IF' /.(CODE+0111) 'THEN' ( <H> MATCH<=/+3  ) ;
       'DO' FETCH;
  SECONDOF : <H> ADD(3:3)<=MEMORY(,$(MADD)),'GOTO'FETCH;
STOP : ;

R; T=3.10/0.93 08.09.46
```

Résultats du traitement de l'unité MODULE

```
karr1
<> R; T=0.37/0.87 09.23.19

start izenter module
EXECUTION BEGINS...
 S:=A ;
  'SI' &ETA(  1) 'ALORS'   (<H> A<=A+-P) ;
  'SI' &ETA(  2) 'ALORS'   (<H> A<=A+P) ;
  'SI' &ETA(  3) 'ALORS'   (<H> M<=M.A) ;
  'SI' &ETA(  3) 'ALORS'   (<H> A<=0) ;
  'SI' &ETA(  4) 'ALORS'   (<H> P<=A) ;
  'SI' &ETA(  5) 'ALORS'   (<H> A<=M) ;
  'SI' &ETA(  5) 'ALORS'   (<H> M<=1) ;
  'SI' &ETA(  6) 'ALORS'   (<H> A<=11) ;
  'SI' &ETA(  7) 'ALORS'   (<H> A<=12) ;
 100000000
 010000000
 001100000
 000010000
 000001100
 000000010
 000000001
R; T=0.30/0.50 09.23.51
```

Résultats du traitement de l'unité CALMAT

```
karrl
R; T=0.42/1.02 08.15.53

start izenter calmat
EXECUTION BEGINS...
<>

    <INCRAD>  'SI'   ADD(8) 'ALORS'   ADD(1:7)<=ADD(1:7)+CARRY ;
 <INCRAD> ADD(8)<=-ADD(8) ;
 <INCRAD2> ADD(1:7)<=ADD(1:7)+CARRY ;
 CARRY(1:6):=ADD(2:7).CARRY(2:7) ;
 CARRY(7):=1 ;
  'POUR'   I = 0 'A'   6  'DEBUT'    'POUR'   J = 0 'A'   6  'DEBUT'   MODULE |I+8.J| (MODULE |I+8+8.
J| (,,;*),MODULE |I+1+8.J| (,,;*),;); 'FIN'   ; 'FIN'    ;
  'POUR'   I = 0 'A'   6  'DEBUT'   MODULE |7+8.I| (MODULE |7+8+8.I| (,,;*),,;); 'FIN'    ;
  'POUR'   I = 0 'A'   6  'DEBUT'   MODULE |I+56| (,MODULE |I+57| (,,;*),,;); 'FIN'    ;
  'POUR'   I = 0 'A'   7  'DEBUT'   MODULE |I+56| (MODULE |I| (,,;*),,;); 'FIN'    ;
  'POUR'   J = 0 'A'   7  'DEBUT'   MODULE |8.J| (,,;S(J)); 'FIN'    ;
  'SI' &ETA(  3) 'ALORS'    ( 'FAIRE'   FETCH) ;
  'SI' &ETA(  1) 'ALORS'   (<H>  'ALLERA'   DECODE) ;
  'SI' &ETA(  4) 'ALORS'   (<H>  'ALLERA'   FETCH) ;
  'SI' &ETA(  2) 'ALORS'   (<H>  'SI'   -CODE(4).-CODE(3).-CODE(2).-CODE(1) 'ALORS'   ( 'POUR'   I =
0 'A'   7  'DEBUT'    'POUR'   J = 0 'A'   7  'DEBUT'    'ALLERA'   ST1 'DE'   MODULE |I+8.J| (,,;),
  'FIN'   , 'FIN'   )) ;
  'SI' &ETA(  2) 'ALORS'   (<H>  'SI'   CODE(4).-CODE(3).-CODE(2).-CODE(1) 'ALORS'   ( 'POUR'   I = 0
 'A'   7  'DEBUT'    'POUR'   J = 0 'A'   7  'DEBUT'    'ALLERA'   ST2 'DE'   MODULE |I+8.J| (,,;),
'FIN'   , 'FIN'   )) ;
  'SI' &ETA(  2) 'ALORS'   (<H>  'SI'   -CODE(4).CODE(3).-CODE(2).-CODE(1) 'ALORS'   ( 'POUR'   I = 0
 'A'   7  'DEBUT'    'POUR'   J = 0 'A'   7  'DEBUT'    'ALLERA'   ST3 'DE'   MODULE |I+8.J| (,,;),
'FIN'   , 'FIN'   )) ;
  'SI' &ETA(  2) 'ALORS'   (<H>  'SI'   CODE(4).CODE(3).-CODE(2).-CODE(1) 'ALORS'   ( 'POUR'   I = 0
'A'   7  'DEBUT'    'POUR'   J = 0 'A'   7  'DEBUT'    'ALLERA'   ST4 'DE'   MODULE |I+8.J| (,,;), '
FIN'   , 'FIN'   )) ;
```

```
'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   -CODE(4).-CODE(3).CODE(2).-CODE(1) 'ALORS'  ( 'POUR'
'A'  7  'DEBUT'   'POUR'   J = 0 'A'  7  'DEBUT'   'ALLERA'   ST5 'DE'   MODULE |1+8.J| (,,;),
'FIN'   , 'FIN'   )) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   CODE(4).-CODE(3).CODE(2).-CODE(1) 'ALORS'  ( 'POUR'   I = 0
'A'  7  'DEBUT'   'POUR'   J = 0 'A'  7  'DEBUT'   'ALLERA'   ST6 'DE'   MODULE |1+8.J| (,,;), '
FIN'   , 'FIN'   )) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   CODE(3).CODE(2).-CODE(1) 'ALORS'  ( 'POUR'   I = 0 'A'  7
'DEBUT'   'POUR'   J = 0 'A'  7  'DEBUT'   'ALLERA'   ST7 'DE'   MODULE |1+8.J| (,,;), 'FIN'   ,
'FIN'   )) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   -CODE(1) 'ALORS'  ( 'POUR'   I = 0 'A'  7  'DEBUT'   'POUR
'   J = 0 'A'  7  'DEBUT'   'ALLERA'   EXECUTE, 'FIN'   , 'FIN'   )) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   -CODE(4).-CODE(3).CODE(2).CODE(1)+MATCH.-CODE(3).CODE(2).CODE
(1) 'ALORS'   ADD(1:4)<=MEMORY(,$(ADD))) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   -CODE(4).-CODE(3).CODE(2).CODE(1)+MATCH.-CODE(3).CODE(2).CODE
(1) 'ALORS'   INCRAD:=1) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   -CODE(4).-CODE(3).CODE(2).CODE(1)+MATCH.-CODE(3).CODE(2).CODE
(1) 'ALORS'   ( 'ALLERA'   SECONDOP)) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   -MATCH.CODE(4).-CODE(3).CODE(2).CODE(1) 'ALORS'   INCRAD2:=1)
;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   -MATCH.CODE(4).-CODE(3).CODE(2).CODE(1) 'ALORS'   ( 'ALLERA'
 FETCH)) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   CODE(4).CODE(3).CODE(2).CODE(1) 'ALORS'   ( 'ALLERA'   STOP))
;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   -CODE(4).-CODE(3).-CODE(2).CODE(1) 'ALORS'   MATCH<=0) ;
 'SI' &ETA( 2) 'ALORS'  (<H>  'SI'   CODE(4).-CODE(3).-CODE(2).CODE(1) 'ALORS'   MATCH<=-MATCH) ;
 'SI' &ETA( 2) 'ALORS'   ( 'SI'   -/+CODE(2:3).CODE(1) 'ALORS'   ( 'FAIRE'   FETCH)) ;
 'SI' &ETA( 3) 'ALORS'   ( 'SI'   -/.(CODE+0111) 'ALORS'   ( 'POUR'   J = 0 'A'  7  'DEBUT'   MOD
ULE |7+8.J| (,MODULE |8.J| (,,;*),;); 'FIN'   )) ;
 'SI' &ETA( 3) 'ALORS'   ( 'SI'   /.(CODE+0111) 'ALORS'   ( 'POUR'   J = 0 'A'  7  'DEBUT'   MODU
LE |7+8.J| (,E(J),;); 'FIN'   )) ;
 'SI' &ETA( 3) 'ALORS'  (<H>  'SI'   /.(CODE+0111) 'ALORS'   MATCH<=/+S) ;
 'SI' &ETA( 1) 'ALORS'  (<H> INCRAD:=1) ;
 'SI' &ETA( 1) 'ALORS'  (<H> CODE<=MEMORY(,$(ADD))) ;
 'SI' &ETA( 4) 'ALORS'  (<H> ADD(5:8)<=MEMORY(,$(NADD))) ;
 'SI' &ETA( 3) 'ALORS'   ( 'POUR'   I = 0 'A'  7  'DEBUT'   'POUR'   J = 0 'A'  7  'DEBUT'   MO
DULE |1+8.J| (,,H;); 'FIN'   ; 'FIN'   ) ;
1110000000000000000000000000
0001111111111111111110000000
0000000000000000000111111000
0000000000000000000000000011
0000000000000000000000000000
R; T=3.75/5.32 09.22.55
```

Texte CASSANDRE de l'unité COMPUTE 1

EDITION

U. S. TEXTE SOURCE

```
00001  'UNITE' COMPUTE1(H,BIN(0:15),BK(0:15),IE(4:15),ACC,ARE;BOU(0:15));
00060    'REGISTRE' SP(0:15,0:15),S(0:15),FBOU(0:15),P(0:15),MEMO(0:15,0:255)
00118    ,FIOVAL,K(0:15);
00134    'SIGNAL' MEM(0:15),TSB(0:3),TSA(0:3),PHB,PHA,P1M,CFOR,OV,SIGMA(0:15)
00194    ,SR1(0:3),CAAF,KYMF,PP(0:15),BUS(0:15),SEL(0:15),B,A,NA(0:7);
00255    'HORLOGEMERE' H;
00258    'HORLOGE' H1;
00262    'ETAT' ETAT(0:255);
00275    'EXTERNE' GEN1('HORLOGE';,,'HORLOGE'),ADD1((0:15),(0:15);(0:15),);
00316                GEN1(H;PHA,PHB,H1);
00335          . <H>
00338                ETAT(33)<=ETAT33,
00354                ETAT(49)<=ETAT49,
00370                ETAT(177)<=ETAT177,
00388                ETAT(161)<=ETAT161,
00406                ETAT(169)<=ETAT169,
00424                ETAT(34)<=ETAT34;
00440                ADD1(P,-*E1(14)&10;PP,);
00462                TSA:=SR1;
00470                TSB:=K(14)&K(11:13);
00489                BOU:=FBOU;
00498                P1M:=-MEM(0)./.-MEM(5:7)+MEM(0)./.MEM(5:8);
00540                NA(0):=CFOR.-K(9).-K(10)+KYMF.MEM(0)./+MEM(9:10).-/.-MEM
00595                (11:14)+KYMF.MEM(0)./.-MEM(9:10).-/.-MEM(11:14);
00643                NA(1):=CFOR+KYMF.-MEM(0)./.MEM(2:4)+KYMF.-MEM(0).(MEM(1)
00698                .-MEM(3)+-/.MEM(2:4)).MEM(1)+KYMF.MEM(0)./.-MEM(9:10).
00752                MEM(1);
```

```
00759                   NA(2):=KYMF.-MEM(0).(MEM(1).-MEM(3)+-/.MEM(2:4)).MEM(2)+
00814                   KYMF.MEM(0)./+MEM(9:10)+KYMF.MEM(0)./.-MEM(9:10).MEM(2);
00870                   NA(3):=KYMF.-MEM(0)./.MEM(2:4)+KYMF.-MEM(0).MEM(3).(MEM(
00925                   1).-MEM(3)+-/.MEM(2:4))+KYMF.MEM(0)./+MEM(9:10).MEM(15)+
00981                   KYMF.MEM(0)./.-MEM(9:10).MEM(3);
01013                   NA(4):=CFOR+KYMF.-MEM(0).(MEM(1).-MEM(3)+-/.MEM(2:4)).
01066                   MEM(4)+KYMF.MEM(0)./+MEM(9:10).MEM(4);
01104                   NA(5):=KYMF.-MEM(0)./.MEM(2:4).MEM(10)+KYMF.-MEM(0).-P1M
01159                   .(MEM(1).-MEM(3)+-/.MEM(2:4))+KYMF.MEM(0)./.-MEM(9:10).-
01215                   P1M;
01219                   NA(6):=KYMF.-MEM(0)./.MEM(2:4).MEM(9)+KYMF.-MEM(0).MEM(1
01274                   ).-MEM(3)+KYMF.MEM(0)./+MEM(9:10).MEM(9)+KYMF.MEM(0)./.-
01330                   MEM(9:10).MEM(9).(-MEM(15)+MEM(1)+-MEM(2))+KYMF.MEM(0)./
01386                   .-MEM(9:10).MEM(15).-MEM(1).MEM(2);
01421                   NA(7):=KYMF.-MEM(0)./.MEM(2:4)+KYMF.-MEM(0).MEM(1).-MEM(
01476                   3)./.-MEM(11:14).-MEM(15).-MEM(10)+KYMF.MEM(0)./+MEM(9:
01531                   10).MEM(10)+KYMF.MEM(0)./.-MEM(9:10);
01568   FETCHP:
01575                   'FAIRE' NOPX;
01581                   'FAIRE' NOPIS;
01588                   'FAIRE' NOPMR;
01595                   'FAIRE' NOPMX;
01602           'SI' PHA 'ALORS'
01607            (  'FAIRE' KYM;
01613                       'FAIRE' KYMO;
01619                       'FAIRE' KYMASP;
01627                       'FAIRE' KYMUS);
01635           'SI' PHB 'ALORS'
01640            (  'FAIRE' NOPRAM;
01649                       'FAIRE' READ;
01655                       'FAIRE' BUSYPP;
01663                       'FAIRE' FBOUYB);
01672              <H1>
01676                   'ALLERA' ETAT($(NA)) ;
01689   ETAT33:
01696                   'FAIRE' NOPX;
01702                   'FAIRE' SELYPP;
01710                   'FAIRE' NOPMR;
01717                   'FAIRE' CAA;
```

```
01722                 'FAIRE' NOPCR;
01729                 'FAIRE' NOPMX;
01736                 'FAIRE' R1;
01740                 'FAIRE' NOPUS;
01747         'SI' PHB 'ALORS'
01752          (  'FAIRE' IMAC;
01759                     'FAIRE' READ;
01765                     'FAIRE' SYPP);
01772            <H1>
01776                 'ALLERA' ETAT($(NA)) ;
01789   ETAT49:
01796                 'FAIRE' NOPX;
01802                 'FAIRE' SELYPP;
01810                 'FAIRE' NOPMR;
01817                 'FAIRE' CAA;
01822                 'FAIRE' NOPCR;
01829                 'FAIRE' NOPMX;
01836                 'FAIRE' R1;
01840                 'FAIRE' NOPUS;
01847         'SI' PHB 'ALORS'
01852          (  'FAIRE' IMAC;
01859                     'FAIRE' READ;
01865                     'FAIRE' SYPP);
01872            <H1>
01876                 'ALLERA' ETAT($(NA)) ;
01889   ETAT177:
01897                 'FAIRE' XYR2;
01903                 'FAIRE' NOPIS;
01910                 'FAIRE' NOPMR;
01917                 'FAIRE' VALRA07;
01926                 'FAIRE' NOPCR;
01933                 'FAIRE' NOPMX;
01940                 'FAIRE' R1;
01944                 'FAIRE' NOPUS;
01951         'SI' PHB 'ALORS'
01956          (  'FAIRE' NOPRAM;
01965                     'FAIRE' READ;
01971                     'FAIRE' BUSYPP;
01979                     'FAIRE' FBOUYB);
01988            <H1>
01992                 'ALLERA' T3AL2 ;
```

```
01909   ETAT161:
02007                   'FAIRE' XYR2;
02013                   'FAIRE' NOPIS;
02020                   'FAIRE' NOPMR;
02027                   'FAIRE' VALRA07;
02036                   'FAIRE' NOPCR;
02043                   'FAIRE' NOPMX;
02050                   'FAIRE' R1;
02054                   'FAIRE' NOPUS;
02061           'SI' PHB 'ALORS'
02066           (  'FAIRE' BUSYPP;
02075                      'FAIRE' FBOUYB;
02083                      'FAIRE' NOPRAM;
02091                      'FAIRE' READ);
02098             <H1>
02102                   'ALLERA' T3AL2 ;
02109   T3AL2:
02115                   'FAIRE' BUSYX;
02122                   'FAIRE' SELYBUS;
02131                   'FAIRE' NOPMR;
02138                   'FAIRE' CAA;
02143                   'FAIRE' NOPCR;
02150                   'FAIRE' NOPMX;
02157                   'FAIRE' R1;
02161                   'FAIRE' NOPUS;
02168           'SI' PHB 'ALORS'
02173           (  'FAIRE' IMAC;
02180                      'FAIRE' READ;
02186                      'FAIRE' SYBUS);
02194   ETAT169:
02202                   'FAIRE' NOPX;
02208                   'FAIRE' NOPIS;
02215                   'FAIRE' NOPMR;
02222                   'FAIRE' VALRA08;
02231                   'FAIRE' NOPCR;
02238                   'FAIRE' NOPMX;
02245                   'FAIRE' R11;
02250                   'FAIRE' NOPUS;
02257           'SI' PHB 'ALORS'
02262           (  'FAIRE' NOPRAM;
```

```
02271                    'FAIRE' READ;
02277                    'FAIRE' BUSYPP;
02285                    'FAIRE' FBOUYB);
02294          <H1>
02298               'ALLERA' T3BLOAD2 ;
02308  T3BLOAD2:
02317               'FAIRE' XYA2R1;
02325               'FAIRE' NOPIS;
02332               'FAIRE' NOPMR;
02339               'FAIRE' VALRA08;
02348               'FAIRE' NOPCR;
02355               'FAIRE' NOPMX;
02362               'FAIRE' R11;
02367               'FAIRE' NOPUS;
02374       'SI' PHB 'ALORS'
02379        (  'FAIRE' NOPRAM;
02388                    'FAIRE' READ;
02394                    'FAIRE' NOPIB);
02402          <H1>
02406               'ALLERA' T3BLOAD3 ;
02416  T3BLOAD3:
02425               'FAIRE' R1YBYX;
02433               'FAIRE' SELYBUS;
02442               'FAIRE' NOPMR;
02449               'FAIRE' CAA;
02454               'FAIRE' NOPCR;
02461               'FAIRE' NOPMX;
02468               'FAIRE' R1;
02472               'FAIRE' NOPUS;
02479       'SI' PHB 'ALORS'
02484        (  'FAIRE' IMAC;
02491                    'FAIRE' READ;
02497                    'FAIRE' SYBUS);
02505          <H1>
02509               'ALLERA' ETAT($(NA)) ;
02522  ETAT34:
02529               'FAIRE' NOPX;
02535               'FAIRE' SELYPP;
02543               'FAIRE' NOPUS;
```

```
02550          'SI' PHB 'ALORS'
02555           (  'FAIRE' NOPRAM;
02564                      'FAIRE' READ;
02570                      'FAIRE' NOPIB);
02578             <H1>
02582                  'ALLERA' T3BLOAD3 ;
02592    NOPUS:
02598                  A:=B;
02602    NOPX:
02607                  A:=B;
02611    NOPIS:
02617                  A:=B;
02621    NOPMR:
02627                  A:=B;
02631    NOPMX:
02637                  A:=B;
02641    NOPRAM:
02648                  A:=B;
02652    KYMO:
02657                  A:=B;
02661    KYMASP:
02668                  A:=B;
02672    KYMUS:
02678                  A:=B;
02682    KYM:
02686               <H>
02689                  KK=MEM,
02695                  FIOVAL<=1;
02704                  KYMF:=1;
02711    READ:
02716                  MEM:=MEMO(,$(SEL));
02734    BUSYPP:
02741                  BUS:=PP;
02748    FBOUYB:
02755               <H>
02758                  FBOU<=BUS;
02767    SELYPP:
02774          'SI' PHB 'ALORS'
02779                    SEL:=PP
02785          'SINON'
02786                       SEL:=-*E1(16);
```

```
02737   CAA:
02801                   CAAF:=1;
02808   NOPCR:
02814                   A:=B;
02818   R1:
02821                   SR1(0):=K(0).K(8);
02838                   SR1(1):=K(5);
02850                   SR1(2):=K(6);
02862                   SR1(3):=K(7);
02874   XYR2:
02879                   ADD1(-*E1(16),SP(,$(TSB));SIGMA,OV);
02913   VALRA07:
02921                   RA(4:7):=0111;
02934   IMAC:
02939                   A:=B;
02943   SYPP:
02948           <H>
02951                   S<=PP;
02956   BUSYX:
02962                   BUS:=SIGMA;
02972   SELYBUS:
02980                   SEL:=BUS;
02988   SYBUS:
02994           <H>
02997                   S<=BUS;
03003   VALRA08:
03011                   RA(4:7):=1000;
03024   R11:
03028                   SR1:=1111;
03037   XYA2R1:
03044                   ADD1(-*E1(14)&10,SP(,$(TSA));SIGMA,OV);
03081   NOPIB:
03087                   A:=B;
03091   R1YBYX:
03098                   BUS:=SIGMA;
03108           <H>
03111                   SP(,$(TSA))<=SIGMA;
```

FIN D'EDITION

Résultats du traitement de l'unité COMPUTE1

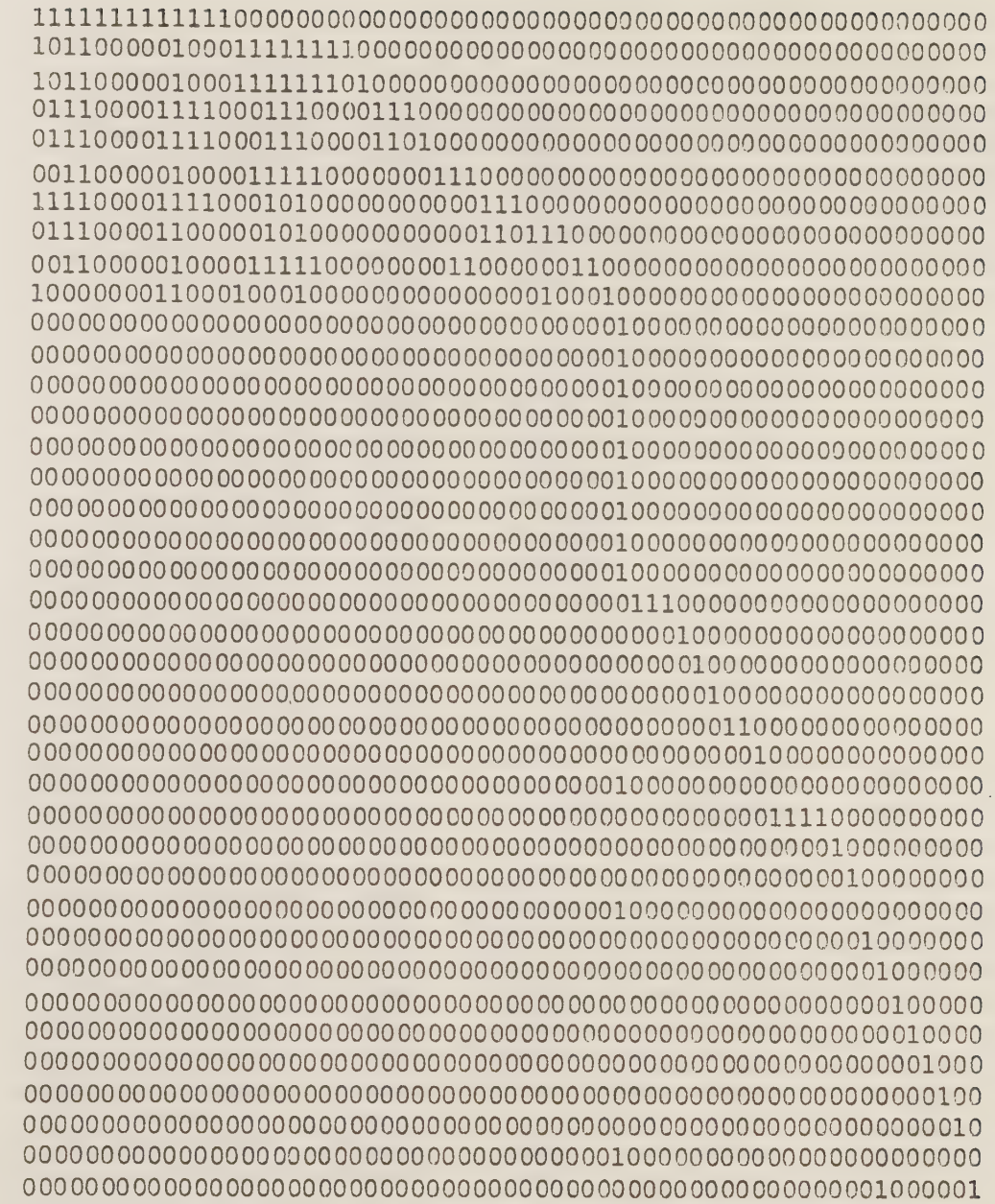
Texte CASSANDRE de sa partie opérative.

```
start izenter compute1
EXECUTION BEGINS...
<> <>

 <H> ETAT(33)<=ETAT33 ;
 <H> ETAT(49)<=ETAT49 ;
 <H> ETAT(177)<=ETAT177 ;
 <H> ETAT(161)<=ETAT161 ;
 <H> ETAT(169)<=ETAT169 ;
 <H> ETAT(34)<=ETAT34 ;
 TSA:=SR1 ;
 TSB:=K(14) & K(11:13) ;
 BOU:=FBOU ;
 P1M:=-MEM(0)./.-MEM(5:7)+MEM(0)./.MEM(5:8) ;
 NA(0):=CFOR.-K(9).-K(10)+KYMF.MEM(0)./+MEM(9:10).-/.-MEM(11:14)+KYMF.MEM(0)./.-MEM(9:10).-/.-MEM(11:
 14) ;
 NA(1):=CFOR+KYMF.-MEM(0)./.MEM(2:4)+-/.MEM(2:4).KYMF.-MEM(0).MEM(1)+MEM(1).-MEM(3).KYMF.-MEM(0).MEM(
 1)+KYMF.MEM(0)./.-MEM(9:10).MEM(1) ;
 NA(2):=-/.MEM(2:4).KYMF.-MEM(0).MEM(2)+MEM(1).-MEM(3).KYMF.-MEM(0).MEM(2)+KYMF.MEM(0)./+MEM(9:10)+KY
 MF.MEM(0)./.-MEM(9:10).MEM(2) ;
 NA(3):=KYMF.-MEM(0)./.MEM(2:4)+-/.MEM(2:4).KYMF.-MEM(0).MEM(3)+MEM(1).-MEM(3).KYMF.-MEM(0).MEM(3)+KY
 MF.MEM(0)./+MEM(9:10).MEM(15)+KYMF.MEM(0)./.-MEM(9:10).MEM(3) ;
  A(4):=CFOR+-/.MEM(2:4).KYMF.-MEM(0).MEM(4)+MEM(1).-MEM(3).KYMF.-MEM(0).MEM(4)+KYMF.MEM(0)./+MEM(9:1
 0).MEM(4) ;
```

```
NA(5):=KYMF.-MEM(0)./.MEM(2:4).MEM(10)+-/.MEM(2:4).KYMF.-MEM(0).-P1M+MEM(1).-MEM(3).KYMF.-MEM(0).-P1
M+KYMF.MEM(0)./.-MEM(9:10).-P1M ;
 A(6):=KYMF.-MEM(0)./.MEM(2:4).MEM(9)+KYMF.-MEM(0).MEM(1).-MEM(3)+KYMF.MEM(0)./+MEM(9:10).MEM(9)+-ME
M(2).KYMF.MEM(0)./.-MEM(9:10).MEM(9)+MEM(1).KYMF.MEM(0)./.-MEM(9:10).MEM(9)+-MEM(15).KYMF.MEM(0)./.-
MEM(9:10).MEM(9)+KYMF.MEM(0)./.-MEM(9:10).MEM(15).-MEM(1).MEM(2) ;
NA(7):=KYMF.-MEM(0)./.MEM(2:4)+KYMF.-MEM(0).MEM(1).-MEM(3)./.-MEM(11:14).-MEM(15).-MEM(10)+KYMF.MEM(
0)./+MEM(9:10).MEM(10)+KYMF.MEM(0)./.-MEM(9:10) ;
CEN1 (P;PHA,PHB,P1) ;
ADD1 (P,-*E1(14) & 10;PP,) ;
 'SI' MICROCOM(1) 'ALORS'    ( 'FAIRE'   NOPX) ;
 'SI' MICROCOM(2) 'ALORS'    ( 'FAIRE'   NOPIS) ;
 'SI' MICROCOM(3) 'ALORS'    ( 'FAIRE'   NOPMR) ;
 'SI' MICROCOM(4) 'ALORS'    ( 'FAIRE'   NOPMX) ;
 'SI' MICROCOM(14) 'ALORS'    ( 'FAIRE'   SELYPP) ;
 'SI' MICROCOM(15) 'ALORS'    ( 'FAIRE'   CAA) ;
 'SI' MICROCOM(16) 'ALORS'    ( 'FAIRE'   NOPCR) ;
 'SI' MICROCOM(17) 'ALORS'    ( 'FAIRE'   R1) ;
 'SI' MICROCOM(18) 'ALORS'    ( 'FAIRE'   NOPUS) ;
 'SI' MICROCOM(23) 'ALORS'    ( 'FAIRE'   XYR2) ;
 'SI' MICROCOM(24) 'ALORS'    ( 'FAIRE'   VALRAO7) ;
 'SI' MICROCOM(27) 'ALORS'    ( 'FAIRE'   BUSYX) ;
 'SI' MICROCOM(28) 'ALORS'    ( 'FAIRE'   SELYBUS) ;
 'SI' MICROCOM(30) 'ALORS'    ( 'FAIRE'   VALRAO8) ;
 'SI' MICROCOM(31) 'ALORS'    ( 'FAIRE'   R11) ;
 'SI' MICROCOM(33) 'ALORS'    ( 'FAIRE'   XYA2R1) ;
 'SI' MICROCOM(36) 'ALORS'    ( 'FAIRE'   R1YBYX) ;
 'SI' MICROCOM(13) 'ALORS'   (<H1>  'ALLERA'   ETAT($(NA))) ;
 'SI' MICROCOM(21) 'ALORS'   (<H1>  'ALLERA'   ETAT($(NA))) ;
 'SI' MICROCOM(22) 'ALORS'   (<H1>  'ALLERA'   ETAT($(NA))) ;
 'SI' MICROCOM(25) 'ALORS'   (<H1>  'ALLERA'   T3AL2) ;
 'SI' MICROCOM(26) 'ALORS'   (<H1>  'ALLERA'   T3AL2) ;
 'SI' MICROCOM(32) 'ALORS'   (<H1>  'ALLERA'   T3BLOAD2) ;
 'SI' MICROCOM(35) 'ALORS'   (<H1>  'ALLERA'   T3BLOAD3) ;
 'SI' MICROCOM(37) 'ALORS'   (<H1>  'ALLERA'   ETAT($(NA))) ;
```

```
'SI' MICROCOM(38) 'ALORS'   (<H1>  'ALLERA'   T3BLOAD3) ;
'SI' MICROCOM(5) 'ALORS'    ( 'SI'   PHA 'ALORS'   ( 'FAIRE'   KYM)) ;
'SI' MICROCOM(6) 'ALORS'    ( 'SI'   PHA 'ALORS'   ( 'FAIRE'   KYMO)) ;
'SI' MICROCOM(7) 'ALORS'    ( 'SI'   PHA 'ALORS'   ( 'FAIRE'   KYMASP)) ;
'SI' MICROCOM(8) 'ALORS'    ( 'SI'   PHA 'ALORS'   ( 'FAIRE'   KYMUS)) ;
'SI' MICROCOM(9) 'ALORS'    ( 'SI'   PHB 'ALORS'   ( 'FAIRE'   NOPRAM)) ;
'SI' MICROCOM(10) 'ALORS'    ( 'SI'   PHB 'ALORS'   ( 'FAIRE'   READ)) ;
'SI' MICROCOM(11) 'ALORS'    ( 'SI'   PHB 'ALORS'   ( 'FAIRE'   BUSYPP)) ;
'SI' MICROCOM(12) 'ALORS'    ( 'SI'   PHB 'ALORS'   ( 'FAIRE'   FBOUYB)) ;
'SI' MICROCOM(19) 'ALORS'    ( 'SI'   PHB 'ALORS'   ( 'FAIRE'   IMAC)) ;
'SI' MICROCOM(20) 'ALORS'    ( 'SI'   PHB 'ALORS'   ( 'FAIRE'   SYPP)) ;
'SI' MICROCOM(29) 'ALORS'    ( 'SI'   PHB 'ALORS'   ( 'FAIRE'   SYBUS)) ;
'SI' MICROCOM(34) 'ALORS'    ( 'SI'   PHB 'ALORS'   ( 'FAIRE'   NOPIB)) ;
'SI' MICROCOM(46) 'ALORS'    ( 'SI'   PHB 'ALORS'   SEL:=PP) ;
'SI' MICROCOM(47) 'ALORS'    ( 'SI'   -PHB 'ALORS'   SEL:=-*E1(16)) ;
'SI' MICROCOM(40) 'ALORS'   (<H> K<=MEM) ;
'SI' MICROCOM(41) 'ALORS'   (<H> FIO AL<=1) ;
'SI' MICROCOM(45) 'ALORS'   (<H> FBOU<=BUS) ;
'SI' MICROCOM(55) 'ALORS'   (<H> S<=PP) ;
'SI' MICROCOM(58) 'ALORS'   (<H> S<=BUS) ;
'SI' MICROCOM(62) 'ALORS'   (<H> SP(,$(TSA))<=SIGMA) ;
'SI' MICROCOM(39) 'ALORS'   A:=B ;
'SI' MICROCOM(42) 'ALORS'   KYMF:=1 ;
'SI' MICROCOM(43) 'ALORS'   MEM:=MEMO(,$(SEL)) ;
'SI' MICROCOM(44) 'ALORS'   BUS:=PP ;
'SI' MICROCOM(48) 'ALORS'   CAAF:=1 ;
'SI' MICROCOM(49) 'ALORS'   SR1(0):=K(0).K(8) ;
'SI' MICROCOM(50) 'ALORS'   SR1(1):=K(5) ;
'SI' MICROCOM(51) 'ALORS'   SR1(2):=K(6) ;
'SI' MICROCOM(52) 'ALORS'   SR1(3):=K(7) ;
'SI' MICROCOM(54) 'ALORS'   NA(4:7):=0111 ;
'SI' MICROCOM(56) 'ALORS'   BUS:=SIGMA ;
'SI' MICROCOM(57) 'ALORS'   SEL:=BUS ;
'SI' MICROCOM(59) 'ALORS'   NA(4:7):=1000 ;
'SI' MICROCOM(60) 'ALORS'   SR1:=1111 ;
'SI' MICROCOM(53) 'ALORS'   ADD1 (-*E1(16),SP(,$(TSB));SIGMA,OV) ;
'SI' MICROCOM(61) 'ALORS'   ADD1 (-* 1(14) & 10,SP(,$(TSA));SIGMA,OV) ;
```



R; T=4.45/5.24 07:42:36

BIBLIOGRAPHIE

1 - ANCEAU F., LIDDELL P., MERMET J., PAYAN C., "CASSANDRE : Conception assistée des systèmes digitaux". Onde électrique. Numéro spécial, Janvier 1969.

2 - ANCEAU F., LIDDELL P., MERMET J., PAYAN C., "A language to describe digital systems, Application to logic design", C.O.I.N.S., Miami, december 18-20 1969

3 - ANCEAU F., COUTURIER B., DOUSSY J., PERRON F., "Compilation et simulation du langage CASSANDRE", chapitre II, rapport final de contrat CRI, Industrialisation de CASSANDRE.

4 - BOGO G., GUYOT A., LUX A., MERMET J., PAYAN C., "CASSANDRE and the computer aided logical systems design", Congrès de l'IFIP, 23-28 août 1971.

5 - BOULAYE G., "La microprogrammation" DUNOD 1971.

6 - BOULAYE G., MERMET J., Editors, "Microprogramming", Hermann 1972.

7 - CHATELIN M., HANCZAKOWSKI A., "Minimisation d'une mémoire morte de microprogramme". Rapport de contrat IMAG-DRME, octobre 1970.

8 - CHATELIN M., GUYOT A., HANCZAKOWSKI A., MENARD G., DE POLIGNAC K., "Application du langage CASSANDRE à la microprogrammation". Rapport de contrat IMAG-DRME, avril 1972.

9 - CHEIN M., "Etude de décomposition d'un réseau. Application à l'écriture des fonctions booléennes en sommes et produits". Thèse docteur-ingénieur, Université de Grenoble.

10 - DAVID, FANTINO, MENARD G., "Outils pour le découpage en module, l'implantation et la documentation automatique des machines digitales décrites en CASSANDRE". Rapport interne ENSIMAG 1973.

11 - GRIFFITHS M., PELTIER P., "Grammar transformation as an aid to compiler production". Etude n° FF2-0057, Mai 1968, Centre scientifique IBM, France.

12 - HARRAND Y., "Traitement des files et des listes", DUNOD, Paris 1967.

13 - IVERSON K.E., "A programming language", J. WILEY and sons, Inc. N.Y. London 1962.

14 - KUNTZMANN J., "Algèbre de Boole", DUNOD, Paris 1965.

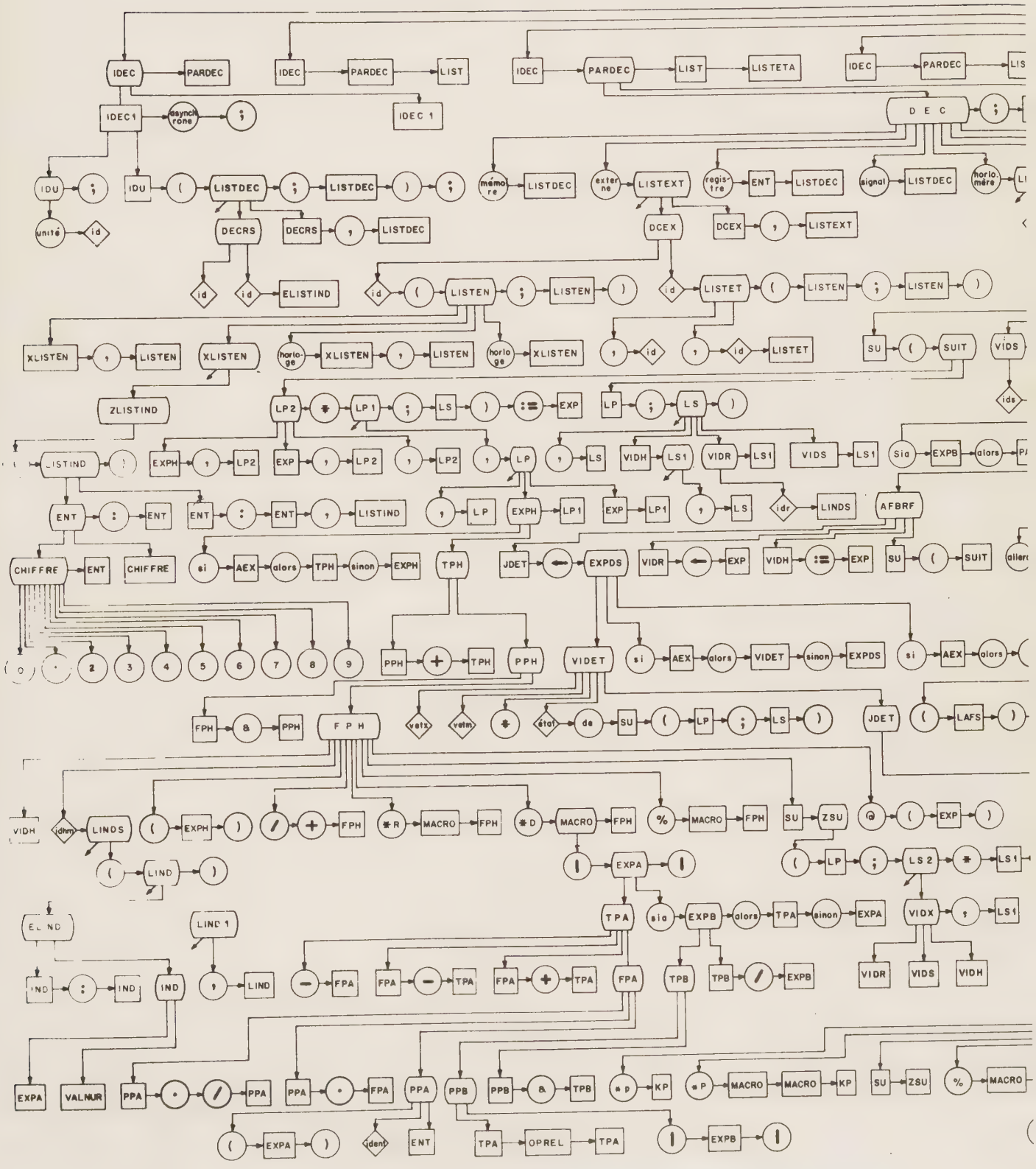
15 - MERMET J., "Définition du langage CASSANDRE", Thèse de docteur-ingénieu Université de Grenoble, mars 1970.

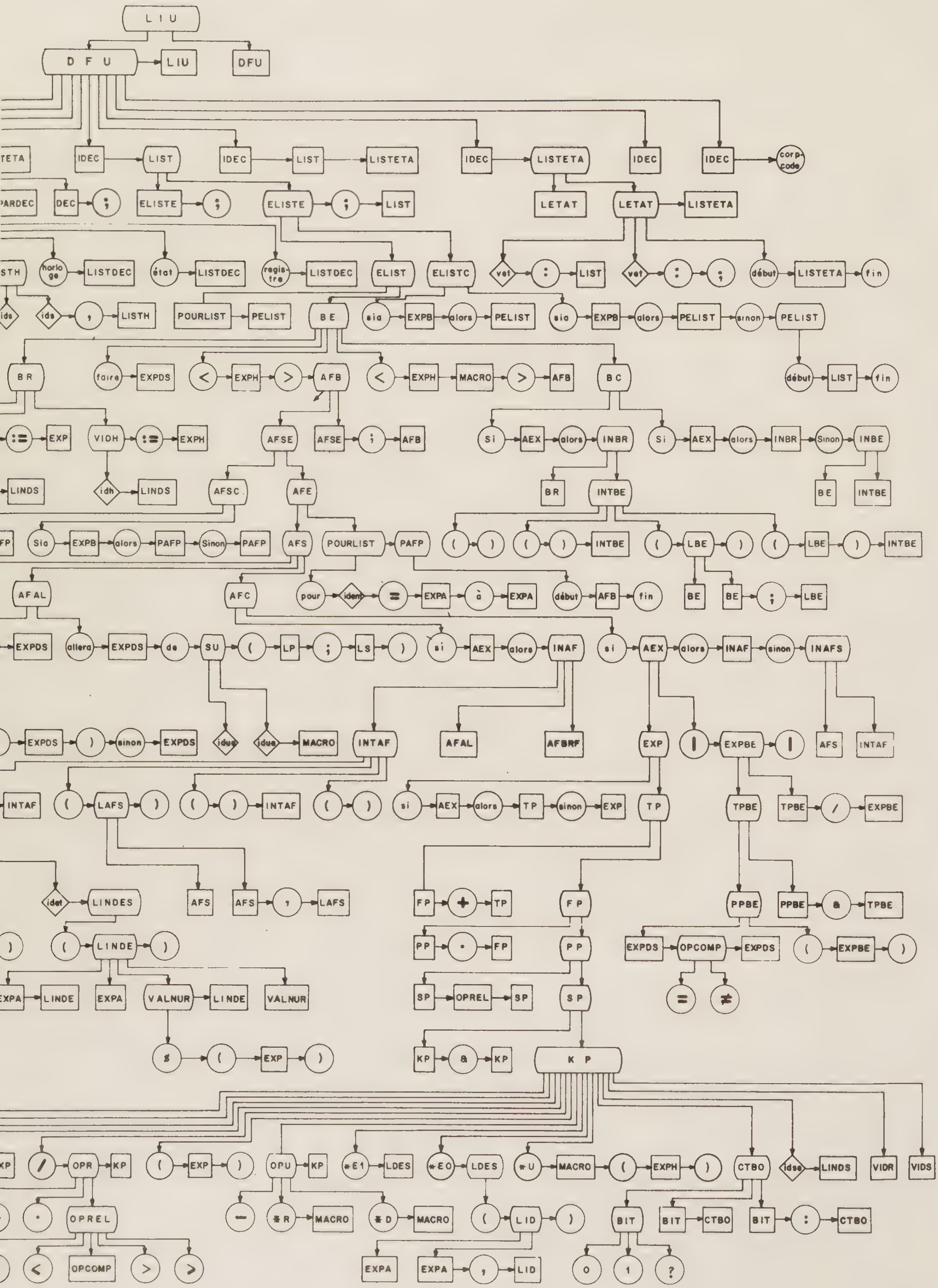
16 - MERMET J., "Etude méthodologique de la conception assistée par ordinate des systèmes logiques", Université de Grenoble, avril 1973.

17 - SIA, "Etude de la microprogrammation. Principe et simulation". Rapport de contrat DRME-IMAG, juin 1969.

ANNEXE I

CARTE SYNTAXIQUE DU LANGAGE CASSANDRE.





L I U
D F U
LIU
DFU
IDEC
LIST
IDEC
LIST
LISTETA
IDEC
LISTETA
IDEC
IDEC
corp-code
DEC
;
ELISTE
;
ELISTE
;
LIST
LETAT
LETAT
LISTETA
horloge
LISTDEC
état
LISTDEC
registre
LISTDEC
ELIST
ELISTC
vet
:
LIST
vet
:
;
début
LISTETA
fin
ids
ids
,
LISTH
POURLIST
PELIST
B E
sia
EXPB
alors
PELIST
sia
EXPB
alors
PELIST
sinon
PELIST
B R
faire
EXPDS
<
EXPH
>
AFB
<
EXPH
MACRO
>
AFB
B C
début
LIST
fin
:=
EXP
VIDH
:=
EXPH
AFSE
AFSE
;
AFB
Si
AEX
alors
INBR
Si
AEX
alors
INBR
Sinon
INBE
LINDS
idh
LINDS
AFSC
AFE
BR
INTBE
BE
INTBE
Sia
EXPB
alors
PAFP
Sinon
PAFP
AFS
POURLIST
PAFP
(
)
(
)
INTBE
(
LBE
)
(
LBE
)
INTBE
AF AL
AFC
pour
ident
=
EXPA
à
EXPA
début
AFB
fin
BE
BE
;
LBE
EXPDS
allera
EXPDS
de
SU
(
LP
;
LS
)
si
AEX
alors
INAF
si
AEX
alors
INAF
sinon
INAFS
EXPDS
)
sinon
EXPDS
idue
idue
MACRO
INTAF
AFAL
AFBRF
EXP
|
EXPBE
|
AFS
INTAF
INTAF
(
LAFS
)
(
)
INTAF
(
)
si
AEX
alors
T P
sinon
EXP
T P
TPBE
TPBE
/
EXPBE
idet
LINDES
AFS
AFS
,
LAFS
FP
+
TP
F P
PPBE
PPBE
&
TPBE
(
LINDE
)
PP
•
FP
P P
EXPDS
OPCOMP
EXPDS
(
EXPBE
)
EXPA
LINDE
EXPA
VALNUR
LINDE
VALNUR
SP
OPREL
SP
S P
=
≠
$
(
EXP
)
KP
&
KP
K P
KP
/
OPR
KP
(
EXP
)
OPU
KP
*E1
LDES
*EO
LDES
*U
MACRO
(
EXPH
)
CTBO
idse
LINDS
VIDR
VIDS
•
OPREL
−
*R
MACRO
*D
MACRO
(
LID
)
BIT
BIT
CTBO
BIT
:
CTBO
<
OPCOMP
>
>
EXPA
EXPA
,
LID
0
1
?

ANNEXE II

CODE INTERNE CASSANDRE

NOM	Valeur hexadécimale	Valeur décimale	REPRESENTATION
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
A	A	10	A
B	B	11	B
C	C	12	C
D	D	13	D
E	E	14	E
F	F	15	F
G	10	16	G
H	11	17	H
I	12	18	I
J	13	19	J
K	14	20	K
L	15	21	L
M	16	22	M
N	17	23	N
O	18	24	O
P	19	25	P
Q	1A	26	Q
R	1B	27	R
S	1C	28	S
T	1D	29	T
U	1E	30	U
V	1F	31	V
W	20	32	W
X	21	33	X

NOM	Valeur hexadécimale	Valeur décimale	REPRESENTATION
Y	22	34	Y
Z	23	35	Z
INCR	24	36	'INCR'
GUILL	25	37	"
ADD	26	38	'ADD'
PH I	27	39	?
DP	28	40	:
PV	29	41	;
BF	2A	42	—
FLG	2B	43	<=
CC	2C	44	|
VALNUM	2D	45	$
DERIU	2E	46	∂
DIF	2F	47	##
INF	30	48	<
SUP	31	49	>
TAU	32	50	%
MOINS	33	51	-
REDUC	34	52	/
PRIM	35	53	.
EG	36	54	=
PLUS	37	55	+
CONC	38	56	&
POINT	39	57	.
VIRG	3A	58	,
RSU	3B	59	–><
PG	3C	60	(
PD	3D	61	)
B3 (Blanc)	3E	62	
BRANCH	3F	63	:=
UNCOUP	40	64	* U
TILDA	41	65	* P
ROT	42	66	* R
DLG	43	67	* D
EPS1	44	68	* E1

NOM	Valeur hexadécimale	Valeur décimale	REPRESENTATION
INDE	45	69	* I
INFG	46	70	- <
SUPG	47	71	> -
A	48	72	'A'
ALLERA	49	73	'ALLERA'
ALORS	4A	74	'ALORS'
ASYNCHRONE	4B	75	'ASYNCHRONE'
CORPCODE	4C	76	'CORPCODE'
DE	4D	77	'DE'
DEBUT	4E	78	'DEBUT'
ETAT	4F	79	'ETAT'
EXTERNE	50	80	'EXTERNE'
FAIRE	51	81	'FAIRE'
FIN	52	82	'FIN'
HORLOGE	53	83	'HORLOGE'
HORLOGEMERE	54	84	'HORLOGEMERE'
POUR	55	85	'POUR'
REGISTRE	56	86	'REGISTRE'
SI	57	87	'SI'
SIA	58	88	'SIA'
SINON	59	89	'SINON'
SIGNAL	5A	90	'SIGNAL'
UNITE	5B	91	'UNITE'
ID	5C	92	Identificateur
IDUD	5D	93	Ident. d'unité
IDUE	5E	94	Ident. d'unité externe
IDS	5F	95	Ident. de signal
IDR	60	96	Ident. de registre
IDH	61	97	Ident. d'horloge
IDHM	62	98	Ident. d'horlogemère
IDET	63	99	Ident. d'état
IDENT	64	100	Ident. d'entier
VET	65	101	Etat
ENT	66	102	Entier
IDSE	67	103	Ident. de signal d'entrée
IVETM	68	104	Ident. d'état multiple

NOM	Valeur hexadécimale	Valeur décimale	REPRESENTATION
VETX	69	105	Etat externe
MEMOIRE	6A	106	'MEMOIRE'
REGETAT	6B	107	'REGETAT'
EQU	6C	108	'EQU'
EPSO	6D	109	* EO
f	6E	110	

ANNEXE III

Les formes types parenthésées et codées décrites ci-après correspondent aux formes types listes décrites au chapitre II. Ce sont les formes sous lesquelles sont stockées, en raison de leur encombrement mémoire moindre, les différentes instructions types retenues au cours du traitement d'une unité CASSANDRE.

On remarquera que ces chaînes codées ne sont pas minimales en nombre d'éléments ; par exemple : les instructions ne commencent pas par le code CASSANDRE les caractérisant, mais celui-ci est précédé systématiquement par le code CASSANDRE correspondant au caractère blanc : 3E. En effet, les formes parenthésées ont été définies de façon à accélérer le passage forme codée, forme liste (et inversement), passage nécessaire fréquemment pour les comparaisons ou les stockages en mémoire.

Les accolades regroupent les différentes formes que l'on peut rencontrer à un niveau donné.

Les noms écrits en majuscules correspondent aux différents éléments types codés décrits.

Les noms écrits en minuscules représentent les pointeurs vers la liste description des identificateurs considérés, à l'exception du nom "entier" qui désigne un entier représenté sur 4 caractères et cadré à droite par exemple : 0012.

Pour les codes hexadécimaux, se reporter au tableau donné en annexe II.

Afin de faciliter la consultation de ces formes codées parenthésées, on a fait figurer le signe typographique des parenthèses ouvertes (ou fermées) plutôt que leur code CASSANDRE 3C (ou 3D) qui figurent réellement en mémoire.

Les formes types parenthésées et codées décrites ci-après correspondent aux formes types listes décrites au chapitre II. Ce sont les formes sous lesquelles sont stockées, en raison de leur encombrement mémoire moindre, les différentes instructions types retenues au cours du traitement d'une unité CASSANDRE.

On remarquera que ces chaînes codées ne sont pas minimales en nombre d'éléments ; par exemple : les instructions ne commencent pas par le code CASSANDRE les caractérisant, mais celui-ci est précédé systématiquement par le code CASSANDRE correspondant au caractère blanc : 3E. En effet, les formes parenthésées ont été définies de façon à accélérer le passage forme codée, forme liste (et inversement), passage nécessaire fréquemment pour les comparaisons ou les stockages en mémoire.

Les accolades regroupent les différentes formes que l'on peut rencontrer à un niveau donné.

Les noms écrits en majuscules correspondent aux différents éléments types codés décrits.

Les noms écrits en minuscules représentent les pointeurs vers la liste description des identificateurs considérés, à l'exception du nom "entier" qui désigne un entier représenté sur 4 caractères et cadré à droite par exemple : 0012.

Pour les codes hexadécimaux, se reporter au tableau donné en annexe II.

Afin de faciliter la consultation de ces formes codées parenthésées, on a fait figurer le signe typographique des parenthèses ouvertes (ou fermées) plutôt que leur code CASSANDRE 3C (ou 3D) qui figurent réellement en mémoire.

```
                ( 3E  51  (EXPDS)
                (
                ( 3E  63  idet    { (IND) } (EXPDS)
                (                 { 3E 3E }
                (
                ( 3E  61  idh     { (IND) } (EXP)
                (                 { 3E 3E }
                (
                ( 3E  60  idr     { (IND) } (EXP)
                (                 { 3E 3E }
                (
INST →          ( 3E  5F  ids     { (IND) } (EXP)
                (                 { 3E 3E }
                (
                ( 3E  5E  idue    { (EXPA)} { (LP) } { (LS) }
                (                 { 3E 3E } { 3E 3E } { 3E 3E }
                (
                ( 3E  49  (EXPDS) { 3E 3E } { (EXPA)} { (LP) } { (LS) }
                (                 { idue  } { 3E 3E } { 3E 3E } { 3E 3E }
                (
                ( 3E  57  (COND)  (INST)
                (
                (
                ( 3E  55  ident   (INST)    (EXPA)    (EXPA)
```

```
COND →  { (EXPB) } { (EXP)  } { (EXPBE)}
        { 3E 3E  } { 3E 3E  } { 3E 3E  }
```

```
LP   →  { (EXP)  }
        { 3E 3E  } ...
```

```
                { idh { (IND) }  }
                {     { 3E 3E }  }
                {                }
                { idr { (IND) }  }
                {     { 3E 3E }  }
                {                }
LS   →          { ids { (IND) }  } ...
                {     { 3E 3E }  }
                {                }
                {  3E 3E         }
```

```
                { 3E 3E                                          }
                { (3E 6E  (EXPA))                                }
                { (3E 2D  (EXP))                                 }
IND  →          { (({ 3E 6E (EXPA) }){ 3E 6E (EXPA) }))         } ...
                {   { 3E 2D (EXP)  }  { 3E 2D (EXP)  }          }
```

ANNEXE IV

```
                 { 3E  3B
                 {
                 { 3E  65  vet
                 {
EXPDS  →         { 3E  68  vetm
                 {
                 { 3E  63  idet  { (IND)  }
                 {               { 3E 3E  }
                 {
                 { 3E  4F  etat    idue  { (EXPA) } { (LP)  } { (LS)  }
                 {                       { 3E 3E  } { 3E 3E } { 3E 3E }
```

```
EXPB  →((3E { 31 } (EXPA) (EXPA))...)...
            { 47 }
            { 36 }
            { 2F }
```

```
EXPBE →((3E { 36 } (EXPDS (EXPDS))...)...
            { 2F }
```

```
EXPA  → (3E { 37 } entier { ident  { ident  }{ ident  }   }
            { 33 }        {   (    { entier }{ entier } ) }...)...
                          {        { (EXPA) }{ (EXPA) }   }
```

```
EXP   →(((  { OPR1 ... { KP1          } }
            {          { 3E 3C (EXP) } }  )...)...)...
            { KP1                      }
```

```
                 { 3E  33
                 {
                 { 3E  32
                 {
OPR1   →         { 3E  34  OPR
                 {
                 { 3E  42  (EXPA)
                 {
                 { 3E  43  (EXPA)
                 {
                 { 3E  41  (EXPA)
```

```
                 { 3E  37
                 {
                 { 3E  39
                 {
                 { 3E  36
                 {
                 { 3E  2F
                 {
OPR    →         { 3E  30
                 {
                 { 3E  31
                 {
                 { 3E  46
                 {
                 { 3E  47
```

KP1 →

```
{ 3E  3E  (EXP)
{
{ 3E  40  (EXPA)   (EXP)
{
{ 3E  6D  { (IND) }
{         { 3E 3E }
{
{ 3E  61  idh      { (IND) }
{                  { 3E 3E }
{
{ 3E  62  idhm     { (IND) }
{                  { 3E 3E }
{
{ 3E  60  idr      { (IND) }
{                  { 3E 3E }
{
{ 3E  5F  ids      { (IND) }
{                  { 3E 3E }
{
{ 3E  67  idse     { (IND) }
{                  { 3E 3E }
{
{ 3E  5E  idue     { (EXPA)} { (LP) } { (LS) } { (LS) }
{                  { 3E 3E } { 3E 3E } { 3E 3E } { 3E 3E }
{
{(3E 00) {3E 00}
{(3E 01) {3E 01}
{(3E 27) {3E 27} ...
{        {3E 28}
```

Cette annexe ne présente qu'un aperçu de l'ensemble des programmes :

- la grammaire, avec les appels de macros et fonctions sémantiques, à partir de laquelle est créé l'analyseur syntaxique.

- quelques unes des fonctions les plus caractéristiques, dont celles mentionnées explicitement au chapitre III.

```
PUT SYNTAX=

80      LIU             DEU 'B3'

81      DEU             'IDUD' LIST LISTETA
                        'IDUD'
                        'IDUD' LISTETA
                        'IDUD' LIST
                        'IDUD' 'CORRCODE'

82      LISTETA         LETAT LISTETA
                        LETAT

83      LETAT           DECETAT COMPTETA 'OPT' SLETAT FINETA
                        DECETAT COMPTETA 'OPT' 'DEBUT' LISTETA 'FIN'

84      DECETAT         'VET' ADPIL DOWN SAVETA
                        ()

85      SLETAT          LIST
                        'PV'

86      LIST            STARTPIL ELISTE DELIM ELEM
                        STARTPIL ELISTE DELIM ELEM LIST

87      DELIM           'PV'

88      ELISTE          ELISTP
                        ELISTC
                        ELIST

89      ELISTP          POURLIST PELIST RANGPLIST

90      POURLIST        'POUR' 'IDENT' ADPIL RANGPIL 'EG' INITPA RANGPIL 'A' INITPA RANGPI

91      PELIST          'DEBUT' INITPIL LIST1

92      LIST1           STARTPIL ELISTE RANGPIL 'VIRG' LIST1
                        STARTPIL ELISTE RANGPIL 'PV' LIST1
                        STARTPIL ELISTE RANGPIL 'PV' 'FIN'
                        STARTPIL ELISTE RANGPIL 'VIRG' 'FIN'
                        STARTPIL ELISTE RANGPIL 'FIN'

93      DELIM1          'VIRG'
                        'PV'
                        ()

94      ELISTC          SIALIST PELIST FCT1
                        SIALIST PELIST FCT1 'SINON' UF FCT2 PELIST FCT1

95      SIALIST         'SIA' INITPIL INITPB CONS1 'ALORS'

96      ELIST           BH
                        BR
                        BC
                        EAFAL
```

```
197                '...' INITPIL ... RANGPIL '...' INITPIL AEC
                   'IND' INITPIL RANGPIL MACRO RANGPIL '...' INITPIL AEC
                   '...' INITPIL ...

198     AEC        LIST2 RANGPIL

199     LIST2      STARTPIL ELIST RANGPIL 'FV' CLAMA
                   STARTPIL ELIST RANGPIL 'VIRG' LIST2
                   STARTPIL ELIST RANGPIL 'FD' CLAMA

200     BF         'IDS' ADPIL LINPS RANGPIL2 'BRANCH' INITP RANCH
                   'IDH' ADPIL LINPS RANGPIL2 'BRANCH' INITP RANCH
                   'IDF' ADPIL LINPS RANGPIL2 'ELC' INITE RANCL
                   'IDET' ADPIL LINPS RANGPIL2 'ELC' INITPOS RANGPIL
                   SUI RANGPIL ZSUB

201     BC         'SI' INSTCOND INITPIL AEX RANGPIL 'ALORS' INEF FCT1PIS
                   'SI' INSTCOND INITPIL AEX RANGPIL 'ALORS' INEF FCT1PIS 'SINON' LP FCT2PIS INBF FCT1PIS RAMASS

202     INBR       INITPIL2 STARTPIL BP RANGPIL RANGPIL
                   INITPIL2 STARTPIL AFAL RANGPIL RANGPIL
                   INITPIL INTRE

203     INBE       INITPIL2 STARTPIL ELIST RANGPIL RANGPIL
                   INITPIL INTRE1

204     INTRE      'PG' INITPIL LBE RANGPIL 'PD' DECCMPCS INTEE1
                   'PG' INITPIL LBE RANGPIL 'PD'
                   'PG' PILVIDE RANGPIL 'PD' DECCMPCS INTBE1
                   'PG' PILVIDE RANGPIL 'PD'

205     LBE        STARTPIL ELIST RANGPIL 'FV' LEE
                   STARTPIL ELIST RANGPIL 'VIRG' LEE
                   STARTPIL ELIST RANGPIL

206     INTRE1     'PG' INITPIL LBE RANGPIL 'PD' INTEE1
                   'PG' INITPIL LBE RANGPIL 'PD'
                   'PG' PILVIDE RANGPIL 'PD' INTEE1
                   'PG' PILVIDE RANGPIL 'PD'

207     SUI        'IDUE' ADPIL RANGPIL PILVIDE
                   'IDUE' ADPIL RANGPIL MACRO

208     AFAL       'ALLERA' INITPOS RANGPIL PILVIDE RANGPIL
                   'ALLERA' INITPOS RANGPIL 'DE' SUI RANGPIL 'FG' INITPIL LP RANGPIL 'FV' INITPIL LS3 RANGPIL 'FD'

209     SU         'IDUE' ADPIL PILVIDE
                   'IDUE' ADPIL MACRO

210     INITZSU    INITPIL ZSU

211     ZSU        'PG' INITPIL LP RANGPIL 'FV' LS RANGPIL 'RSL' LS1 RANGPIL 'PD'

212     MACRO      'CC' INITPA 'CC'

213     AEX        INITPIL LITTP CENS2
                   INITPIL 'CC' INITPR 'CC' CENS2
```

```
ZSUB            'PG' INITPIL SUIT RANGPIL SUIT1
                'PG' 'RSU' PILVIDE FINZSU

SUIT            PILVIDE
                INITP

SUIT1           'VIRG' SUIT2
                'PV' RANGPIL LS2 RANGPIL 'PD'

SUIT2           SUIT RANGPIL SUIT1
                'RSU' FINZSU

FINZSU          RANGPIL LP1 RANGPIL 'PV' LS2 RANGPIL 'PD' 'BRANCH' INITP SIGNU

OPP             'PLUS'
                'POINT'
                'SUP'
                'INF'
                'SUPG'
                'INFG'
                'DIF'
                'EG'

LINDS           'PG' INITPIL LIND 'PD'
                PILVIDE

VALNUR          INITPIL CODPIL 'VALNUM' RANGPIL 'PC' INITP RANGPIL 'PD'

EXPAR           INITPIL INITPA CONSPA

LIND            LIND1 RANGPIL LIND2

LIND1           PILVIDE
                EXPAR
                VALNUR
                EXPAR 'OPT' INITPIL PERM RANGPIL EXPAR RANGPIL
                EXPAR 'OPT' INITPIL PERM RANGPIL VALNUR RANGPIL
                VALNUR 'OPT' INITPIL PERM RANGPIL VALNUR RANGPIL
                VALNUR 'OPT' INITPIL PERM RANGPIL EXPAR RANGPIL

LIND2           'VIRG' LIND
                ()

INITPDS         INITPIL EXPDS

EXPDS           EXPDSC
                CODPIL RANGPIL EXPDSNC RANGPIL

EXPDSNC         'VETM' ADPIL
                'VET' ADPIL
                'PSU' RANGETA
                'VETX' ADPIL
                'IDET' ADPIL RANGPIL LINDS
                'ETAT' ADPIL RANGPIL 'DE' SU1 RANGPIL 'PG' INITPIL LP RANGPIL 'PV' INITPIL LS3 'P

EXPDSC          'SI' AEX 'ALORS' INITPIL EXPDS1 FCT3 'SINON' INITPDS FCT4

EXPDS1          EXPDS
```

```
                        'PG' EXPDS 'PD'

231    INITPB           INITPIL EXPB

232    EXPB             INITPIL2 TPB TRITUR 'REDUC' EXPB
                        INITPIL2 TPB TRITUR

233    TPB              PPB 'CONC' TPB
                        PPB

234    PPB              'CC' INITPB RANGPIL 'CC'
                        INITPA OPREL INITPA RANGERIEN

235    OPREL            'SUP' S2SUP COMO
                        'INF' S2SUP COM1
                        'SUPG' S2SUPG COMO
                        'INFG' S2SUPG COM1
                        OPCOMP COMO

236    OPCOMP           'EG' S2EG
                        'DIF' S2DIF

237    INITPBE          INITPIL EXPBE

238    EXPBE            INITPIL2 TPBE TRITUR 'REDUC' EXPBE
                        INITPIL2 TPBE TRITUR

239    TPBE             PPBE 'CONC' TPBE
                        PPBE

240    PPBE             'PG' INITPBE RANGPIL 'PD'
                        INITPDS OPCOMP INITPDS RANGERIEN

241    INITPA           INITPIL EXPA

242    EXPA             INITPIL SIMPLIF TPA APPL3 INFBIG MIXTE APPL5 FIGNOL
                        'SIA' INITPIL INITPB CONS1 'ALORS' INITPA FCT3 'SINON' INITPA FC

243    TPA              SIGNPLUS TPA1
                        'MOINS' PPA VOIRMOINS

244    TPA1             FPA APPL1 TRITUR PLACE
                        FPA APPL1 TRITUR PLACE 'PLUS' SIGNPLUS TPA1
                        FPA APPL1 TRITUR PLACE 'MOINS' SIGNMOINS TPA1

245    FPA              PPA 'POINT' PROD FPA
                        PPA 'POINT' 'REDUC' PPA SUITDIV
                        PPA PROD

246    PPA              'ENT' APPIL1
                        'IDENT' APPIL
                        'PG' INITPA2 'PD'

247    INITPA2          INITPIL EXPA2

248    EXPA2            INITPIL SIMPLIF TPA2 APPL3 INFBIG MIXTE APPL5 FIGNOL
                        'SIA' INITPIL INITPB CONS1 'ALORS' INITPA FCT3 'SINON' INITPA FC
```

```
249       TPA2        SIGNPLUS TPA12
                      'MOINS' PPA VOIGNMOINS

250       TPA12       PPA2 APPL1 TITUR PLACE
                      PPA2 APPL1 TITUR PLACE 'PLUS' SIGNPLUS TPA1
                      PPA2 APPL1 TITUR PLACE 'MOINS' SIGNMOINS TPA1

251       PPA2        PPA PP??
                      PPA 'POINT' 'REFLC' PPA SLITPIV DIV1
                      PPA 'POINT' PREF FPA

252       INITPH      INITPIL EXPH

253       EXPH        INITPIL4 INITPIL SIMPLIF TPH APPL3 IMPRIQ MIXTE APPL6 FIGNCL
                      'SI' AEX 'ALORS' INITPH FCT3 'SINON' INITPH FCT4

254       TPH         INITPIL4 SP SETSP PERM APPL7 TFITUR PLACE 'PLUS' INITPIL4 TPH
                      INITPIL4 SP SETSP PERM APPL7 TFITUR PLACE

255       INITP       INITPIL EXP

256       EXP         INITPIL4 INITPIL SIMPLIF TP APPL3 IMPRIQ MIXTE APPL6 FIGNCL
                      'SI' AEX 'ALORS' INITP FCT3 'SINON' INITP FCT4

257       TP          FP PERM APPL7 TFITUR PLACE
                      FP PERM APPL7 TFITUR PLACE 'PLUS' INITPIL4 TP

258       FP          INITPIL4 PP 'POINT' FP
                      INITPIL4 FP

259       PP          SP UP4 OPPIL INITPIL4 SP ECRCCL
                      SP SETSP

260       SP          INITPIL COPPIL KP 'CONC' SP
                      INITPIL COPPIL KP SPCONS
                      FAUKP SETFACT 'CONC' SP
                      FAUKP SETFACT SPCONS

261       FAUKP       'PG' INITP 'PD'

262       KP          KP1 FEEL RANGPIL
                      KP2 COPPIL KP3
                      SU FEEL INITZSI ACCRCPIL RANGPIL

263       KP1         'UNCCLP' MACRO 'PG' INITP 'PD'
                      'DERIV' 'PG' INITP 'PD'
                      IDX
                      'INDE' LINDS
                      'EPS' LINDS
                      CTRO

264       KP2         'MOINS'
                      'TAU' MACRO
                      'NOTS' MACRO
                      'DLG' MACRO
                      'TILDA' UNDRIY
                      'TILDA' MACRO MACRO
                      'PRODUC' COPPIL CPF
```

```
265     KP3          KP
                     FAUKP DEV

266     CTE1         BIT CODPIL CTRC
                     BIT
                     BIT CODPIL 'PART' CODPIL CTRC

267     BIT          'NP'
                     'NL'
                     'PHI'

268     LP           INITP RANGPIL LPSUIT
                     PILVIDE RANGPIL LPSUIT

269     LPSUIT       'VIRG' LP
                     ()

270     LP1          'VIRG' INITPIL LP
                     PILVIDE

271     LS           INITPIL INITPIL IDX RANGPIL2 RANGPIL 'VIRG' LSSUIT
                     INITPIL PILVIDE RANGPIL 'VIRG' LSSUIT
                     PILVIDE

272     LSSUIT       PILVIDE RANGPIL 'VIRG' LSSUIT
                     INITPIL IDX RANGPIL2 RANGPIL 'VIRG' LSSUIT
                     ()

273     LS2          INITPIL INITPIL IDX RANGPIL2 RANGPIL LS2SUIT
                     INITPIL PILVIDE RANGPIL 'VIRG' LS2SUIT2
                     PILVIDE

274     LS2SUIT      'VIRG' PILVIDE RANGPIL LS2SUIT
                     'VIRG' INITPIL IDX RANGPIL2 RANGPIL LS2SUIT
                     ()

275     LS2SUIT2     INITPIL IDX RANGPIL2 RANGPIL LS2SUIT
                     PILVIDE RANGPIL LS2SUIT

276     LS1          'VIRG' LS2
                     PILVIDE

277     LS3          PILVIDE RANGPIL LS3SUIT
                     INITPIL IDX RANGPIL2 RANGPIL LS3SUIT

278     LS3SUIT      'VIRG' LS3
                     ()

279     IDX          'IDSF' ADPIL LINDS
                     'IDHM' ADPIL LINDS
                     'IDS' ADPIL LINDS
                     'IDR' ADPIL LINDS
                     'IDH' ADPIL LINDS
                     'IDET' ADPIL LINDS
```

CROSS REFERENCE TABLE * FUNCTIONS

1	IMPRIM	242	248	253	256		
2	MIXTE	242	248	253	256		
3	FIGNOL	242	248	253	256		
4	PLACE	244	250	254	257		
5	SIGNPLUS	243	244	249	250		
6	SIGNMOINS	244	250				
7	PROD	245	251				
8	TRITUR	232	238	244	250	254	257
9	RANGERIEN	234	240				
10	SUITDIV	245	251				
11	VOIRMOINS	243	249				
12	FCT3	229	242	248	253	256	
13	FCT4	229	242	248	253	256	
14	SETFACT	260					
15	FEEL	262					
16	SPCONS	260					
17	SETSP	254	259				
18	EGBOOL	259					
19	DEV	265					
20	RANGPLIST	189					
21	RANGU	200					
22	SIGNU	218					
23	DECOMPOS	204					
24	ELEM	186					
25	FCT1	194					
26	FCT2	194					
27	FCT1BIS	201					
28	FCT2BIS	201					
33	FINETA	183					

CROSS REFERENCE TABLE * MACRO

34	S2SUP	235
35	S2SUPG	235
36	S2DIF	236
37	S2EG	236
38	COM0	235
39	COM1	235
40	APPIL	184 190 200 207 209 228 246 279
41	INITPIL	191 195 197 201 202 203 204 206 208 210 211 213 214 220 221 22 224 226 228 229 231 237 241 242 247 248 252 253 255 256 260 27 271 272 273 274 275 277
42	INITPIL2	202 203 232 238
43	INITPIL4	253 254 256 257 258 259
44	STARTPIL	186 192 199 202 203 205
45	RANGPIL	190 192 197 198 199 200 201 202 203 204 205 206 207 208 211 21 216 217 218 221 223 224 227 228 234 240 262 268 271 272 273 27 275 277
46	COOPIL	221 227 260 262 264 266
47	ACCROPIL	262
48	PILVIDE	197 204 206 207 208 209 214 215 220 224 268 270 271 272 273 27 275 276 277
49	PERM	224 254 257
50	UP	194 201
51	UP4	259
52	CONSRA	222
53	APPL1	244 250
54	SIMPLIF	242 248 253 256
55	APPL3	242 249 253 256
56	RENUA	199
57	APPL5	242 249
58	APPL6	253 256

59	APPL7	254	257					
60	CONS1	195	242	248				
61	CONS2	213						
62	CONS3	213						
63	INSTCOND	201						
64	RAMASS	201						
65	COMPTETA	183						
66	DOWN	184						
67	UNDEUX	264						
68	DIV1	251						
69	RANGPIL2	200	271	272	273	274	275	277
70	SAVETA	184						
71	RANGETA	228						
72	APPIL1	246						

```
                                                     259                ROUTINE RECCMP
[illegible] D2?? [illegible] ?052 [illegible] 00052  272                MVC    LIST1(2),ANIF(TERMAREA
                                                     273                DECAP 1
                                                     276                CAR    0(CONSICNT),WORK1
[illegible] D2?? [illegible] 1000 [illegible] [illegible]  281          MVC    EXP(2),0(WORK1)
[illegible] 4144 ?004                       00004    282                LA     CONSICNT,4(CONSICNT)
[illegible] 5?4? [illegible]                002?8    283                ST     CONSICNT,FONFIL
                                                     284                DECAP 1
                                                     287                CAR    0(WORK1),WORK1
                                                     292                CAR    2(WORK1),WORK1
                                                     297                CAR    0(WORK1),WORK1
?05? D2?? [illegible] 1002 00??? 00002               302  TERM1         MVC    TERM(2),2(WORK1)
                                                     303                CAR    0(WORK1),WORK1
??72 D2?? [illegible] 1002 00??0 00002               308  FACT1         MVC    FACT(2),2(WORK1)
                                                     309                CAR    0(WORK1),WORK1
[illegible] D2?? 8?04 1002 00??4 00002               314                MVC    SAVE1(2),2(WORK1)
[illegible] D2?? 1002 ?052 00002 00052               315                MVC    2(2,WORK1),ANIF(TERMARE
??8? 1???                                            316                LR     WORK2,WORK1
                                                     317                PUSH   WORK2,F
                                                     329                PUSH   SAVE1
??84 45?? ?2?8                              002?8    337                BAL    FARMACD,STOCKE
                                                     338                IF     F,FACT,NE,ANIF(TERMARE
                                                     342                CAR    FACT,WORK1
??54 47?? ?072                              00072    347                B      FACT1
                                                     349                ENDIF ELSE
                                                     351                IF     F,TERM,NE,ANIF(TERMARE
                                                     355                CAR    TERM,WORK1
[illegible] 47?? ?05?                       0005?    360                B      TERM1
                                                     361                ENDIF
                                                     363                ENDELSE
                                                     365                DECAP 6
[illegible] 50?? [illegible]                0092?    378                ST     CONSICNT,SOMFIL
[illegible] D2?? 8??? ?9?? 00??? 009??               379  NEWCOND       MVC    LIST(2),EXP
[illegible] 4144 ?00?                       0000?    380                LA     CONSICNT,12(CONSICNT)
                                                     381                CALL   RECPIE
                                                     384                DECAP 6
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                    397          SNOC   LIST,LIST1
                                    421 CONCAT   IF     H,10(CONSICNT),NE,12(CONSICNT)
0001C2 48A4 000A            0000A   426          LH     WORK1,10(CONSICNT)
0001C6 4111 0001            00001   427          LA     WORK1,1(WORK1)
0001CA D201 8056 4006 00056 00006   428          MVC    ATENT+2(2),6(CONSICNT)
0001D0 D201 8054 4008 00054 00008   429          MVC    ATENT(2),8(CONSICNT)
0001D6 5820 8054            00054   430          L      WORK2,ATENT
0001DA 4012 0000            00000   431          STH    WORK1,0(WORK2)
0001DE 4014 000A            0000A   432          STH    WORK1,10(CONSICNT)
                                    433          ENDIF ELSE
                                    436          IF     H,4(CONSICNT),NE,ANIF(TERMAREA)
                                    440          CAR    4(CONSICNT),WORK1
000208 D201 4004 1002 00004 00002   445          MVC    4(2,CONSICNT),2(WORK1)
00020E D201 8056 4000 00056 00000   446          MVC    ATENT+2(2),0(CONSICNT)
000214 D201 8054 4002 00054 00002   447          MVC    ATENT(2),2(CONSICNT)
00021A 5820 8054            00054   448          L      WORK2,ATENT
00021E D201 2000 1000 00000 00000   449          MVC    0(2,WORK2),0(WORK1)
000224 45E0 C2A8            002A8   450          BAL    EARNADE,STOCKE
                                    451          DECAP 6
                                    464          ENDIF ELSE
000244 5840 C9D8            009D8   467          L      CONSICNT,FONPIL
                                    468          DECAP 2
000250 D201 8054 4000 00054 00000   473          MVC    SAVE1(2),0(CONSICNT)
000256 4144 0002            00002   474          LA     CONSICNT,2(CONSICNT)
                                    475          CALLRECL SAVE1
                                    479          DECAF 1
000266 D201 4000 80CE 00000 000CE   482          MVC    0(2,CONSICNT),LIST1
00026C 4144 0002            00002   483          LA     CONSICNT,2(CONSICNT)
                                    484          EXIT
                                    486          ENDELSE
                                    488          ENDELSE
                                    490          IF     F,CONSICNT,NE,FONPIL
                                    493          DECAF 7
000298 47F0 C13C            0013C   508          B      CONCAT
                                    509          ENDIF ELSE
0002A0 5840 C9DC            009DC   512          L      CONSICNT,SONPIL
0002A4 47F0 C138            00138   513          B      NEWCONC
                                    514          ENDELSE
                                    516 STOCKE   CAR    0(WORK1),WORK1
                                    521          CAR    2(WORK1),WORK1
                                    526          WHILE H,0(WORK1),GT,0
                                    531          CAR    2(WORK1),WORK1
                                    536          ENDWHILE
0002F8 D201 C9D2 1000 009D2 00000   539          MVC    IDENT1(2),0(WORK1)
                                    540          CAR    2(WORK1),WORK1
                                    545          IF     H,0(WORK1),NE,VIDE
                                    549          CAR    0(WORK1),WORK1
                                    554          CAR    0(WORK1),WORK2
                                    559          IF     H,0(WORK2),EQ,ARIT
                                    563          CAR    2(WORK2),WORK1
                                    568          CAR    0(WORK1),WORK1
                                    573          CAR    0(WORK1),WORK1
                                    578          CAR    2(WORK1),WORK1
000376 D201 8054 1000 00054 00000   583          MVC    SAVE1(2),0(WORK1)
```

```
LOC  OBJECT CODE     LINE1 ADDR2  STMT   SOURCE STATEMENT

                                   504          PUSH  WORK1,F
                                   506          PUSH  SAVE1
                                   604          PUSH  SAVE1
                                   612          ENDIF ELSE
                                   615          MVC   BCSAVE2(2),2(WORK2)
                                   616           IF    F,0(WORK2),EQ,VIDE
                                   620           CAR   IDENT1,WORK3
                                   625           CAR   2(WORK3),WORK3
                                   630           CAR   0(WORK3),WORK3
                                   635           MVC   LIST(2),ANIF(TERMARE
                                   636           CONS  0(WORK3),LIST
                                   653           CONS  PLUS,LIST
                                   670           MVC   SUITE(2),ANIF(TERMAR
                                   671           CONS  LIST,SUITE
                                   688           MVC   LIST(2),ANIF(TERMARE
                                   689           CONS  SUITE,LIST
                                   706           CONS  ABIT,LIST
                                   723           MVC   SUITE(2),ANIF(TERMAR
                                   724           CONS  LIST,SUITE
                                   741           MVC   0(2,WORK1),SUITE
                                   742           ENDIF ELSE
                                   745           MVC   0(2,WORK1),0(WORK2)
                                   746           ENDELSE
                                   748           CAR   0(WORK1),WORK1
                                   753           CAR   2(WORK1),WORK1
                                   758           CAR   0(WORK1),WORK1
                                   763           CAR   0(WORK1),WORK1
                                   768           CAR   2(WORK1),WORK1
                                   773          MVC   SAVE1(2),0(WORK1)
                                   774           PUSH  WORK1,F
                                   776           PUSH  SAVE1
                                   784           CAR    BCSAVE2,WORK1
                                   789           IF    F,0(WORK1),EQ,VIDE
                                   313          CAR   IDENT1,WORK3
                                   808          CAR   2(WORK3),WORK3
                                   813          CAR   0(WORK3),WORK3
                                   818          CAR   2(WORK3),WORK3
                                   823          PUSH  0(WORK3)
                                   831          ENDIF ELSE
                                   834          CAR   0(WORK1),WORK1
                                   839          CAR   2(WORK1),WORK1
                                   844          CAR   0(WORK1),WORK1
                                   849          CAR   0(WORK1),WORK1
                                   854          CAR   2(WORK1),WORK1
                                   859          PUSH  0(WORK1)
                                   867          ST    PARMADC,ATENT
                                   868          ST    CONSTCNT,ATENT1
                                   869          L     CONSTCNT,SAVETL
                                   870          LA    CONSTCNT,3(CONSTCNT)
                                   871          CALLFCOL BCSAVE2
                                   875          L     CONSTCNT,ATENT1
                                   876          L     PARMADC,ATENT
                                   877          ENDELSE
                                   878          ENDELSE
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                   881          ENDIF ELSE              IND VIDE
                                   884          CAR   IDENT1,WORK3
                                   889          CAR   2(WORK3),WORK3
                                   894          CAR   0(WORK3),WORK3
                                   899          CAR   2(WORK3),WORK2
000744 D201 8000 8052 00000 00052  904          MVC   LIST(2),ANIF(TERMAREA)
                                   905          CONS  0(WORK3),LIST
                                   922          CAR   LIST,WORK3
000802 D201 8004 E000 00004 00000  927          MVC   SAVE1(2),0(WORK3)
                                   928          PUSH  WORK3,F
                                   940          PUSH  SAVE1
                                   948          PUSH  0(WORK2)
                                   956          CONS  PLUS,LIST
000842 D201 8004 8052 00004 00052  973          MVC   SUITE(2),ANIF(TERMAREA)
                                   974          CONS  LIST,SUITE
000874 D201 8000 8052 00000 00052  991          MVC   LIST(2),ANIF(TERMAREA)
                                   992          CONS  SUITE,LIST
                                  1009          CONS ARIT,LIST
000874 D201 8004 8052 00004 00052 1026          MVC   SUITE(2),ANIF(TERMAREA)
                                  1027          CONS  LIST,SUITE
000898 D201 1000 8004 00000 00004 1044          MVC   0(2,WORK1),SUITE
                                  1045          ENDSLSE
000902 07FE                       1047          BR    R#ERMADE
                                  1048          EXIT
000908 3E37                       1050 PLUS     DC    X'3E37'
00090A 3E68                       1051 ARIT     DC    X'3E68'
00090C                            1052 EXP      DS    H
00090E                            1053 TERM     DS    H
000910                            1054 FACT     DS    H
000912                            1055 IDENT1   DS    F
000916                            1056 BORNE2   DS    H
000918                            1057 EXPRIL   DS    F
00091C                            1058 SIMPIL   DS    F
                                  1059          RETURN
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                   1100          ROUTINE FECRCOL
00A4E 5810 3160             00160  1103          L      WORK1,ASIMPL
00A52 9201 1000       00000        1104          MVI    0(WORK1),1
                                   1105          DECAF 2
00A56 9520 4001       00001        1110          CLI    1(CONSICNT),X'2F'
00A5A 4770 C02C             00A7A  1111          BNE    FC1
00A5E 4144 0004             00004  1112          LA     CONSICNT,4(CONSICNT)
                                   1113          CALL   &NEGSP
                                   1116          DECAF 2
00A74 4700 C028             00A76  1121          B      FC4
00A7A 9526 4001       00001        1122 FC1      CLI    1(CONSICNT),X'36'
00A7E 4780 C028             00A76  1123          BE     FC4
                                   1124          DECAF 4
00A82 9500 4008       00003        1133          CLI    11(CONSICNT),0
00A86 4700 C05E             00AAC  1134          BE     FC2
00A8A D207 4014 4000 00014 00000   1135          MVC    20(8,CONSICNT),0(CONSICNT)
00A90 D207 4000 400C 00000 0000C   1136          MVC    0(8,CONSICNT),12(CONSICNT)
00A96 D207 400C 4014 0000C 00014   1137          MVC    12(8,CONSICNT),20(CONSICNT)
00A9C 9531 4000       00000        1138 FC2      CLI    9(CONSICNT),X'31'
00AA0 4780 C06E             00ABC  1139          BE     FC3
00AA4 4570 C000             0030F  1140          BAL    PARMADC,CREXP
                                   1141          EXIT
                                   1143 FC3      CALL   &SETSP
00AAC 4144 0002             00002  1146          LA     CONSICNT,2(CONSICNT)
00AC4 D207 4000 400C 00000 0000C   1147          MVC    0(8,CONSICNT),12(CONSICNT)
                                   1148          CALL   &NEGSP
                                   1151          CALL   &SETSP
                                   1154          EXIT
                                   1156 FC4      DECAP 4
00A06 D207 C000 4000 00C3C 00000   1156          MVC    SAUVA(8),0(CONSICNT)
00A0C D207 C15A 400C 00C48 0000C   1157          MVC    SAUVB(8),12(CONSICNT)
00AE2 4570 C000             0030F  1158          BAL    PARMADC,CREXP
00AE6 4144 0002             00002  1159          LA     CONSICNT,2(CONSICNT)
00AEA D207 4000 C15A 00000 00C43   1170          MVC    0(8,CONSICNT),SAUVB
00AF0 D207 400C C155 0000C 00C3C   1171          MVC    12(8,CONSICNT),SAUVA
00AF6 4570 C000             0030F  1172          BAL    PARMADC,CREXP
                                   1173          EXIT
00AFA 4144 000C             0000C  1175 CREXP    LA     CONSICNT,12(CONSICNT)
00AFE 5070 C156             00C44  1176          ST     PARMADC,SAUVAD
                                   1177          CALL   &NEGSP
                                   1180          DECAF 6
                                   1193          IF     H,12(CONSICNT),EQ,ANIF(TERMAREA)
00B2A D201 3000 4002 0000C 00002   1197          MVC    LIST(2),2(CONSICNT)
00B2E 4144 0014             00014  1198          LA     CONSICNT,20(CONSICNT)
                                   1199          CALL   &COPIE
                                   1212          DECAP 10
00B74 D201 4002 8000 00002 0000C   1223          MVC    2(2,CONSICNT),LIST
                                   1224          ACCRECHE 2(CONSICNT),14(CONSICNT)
                                   1236          ENDIF ELSE
00B84 D201 3000 4000 0000C 00000   1239          MVC    LIST(2),0(CONSICNT)
00B8A 4144 0014             00014  1240          LA     CONSICNT,20(CONSICNT)
                                   1241          CALL   &COPIE
                                   1244          DECAF 10
00BDA D201 4000 8000 00000 0000C   1265          MVC    0(2,CONSICNT),LIST
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                    1266           CONS  12(CONSICNT),0(CCNSICNT)
                                    1283           ENDELSE
000032 0201 4016 8012 00006 00012   1285           MVC   6(2,CONSICNT),LIMITE
000038 4144 0012              00012 1286           LA    CCNSICNT,2(CCNSICNT)
                                    1287           CALL  &MIXTE
                                    1290           CALL  &SETSP
000034 5850 C1E6              00044 1293           L     R/RMADD,SAUVAD
000038 47FE 0000              00000 1294           B     0(PARMADD)
000030                              1295 SAUVA     DS    2F
000044                              1296 SAUVAD    DS    F
000048                              1297 SAUVB     DS    2F
                                    1298           RETURN
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                  3649          ROUTINE &IMBBIO
                                  3652          DECAR 1
                                  3655          IF    H,0(CONSTCNT),EQ,ANIF(TERMAREA)
002796 47F0 C2F0              02A66 3659          B     PASINB
                                  3660          ENDIF
                                  3662          CAR   0(CONSTCNT),WORK3
002744 D201 805A 6000 0005A 00000 3667          MVC   LISTSI(2),0(WORK3)
                                  3668          IF    H,2(WORK3),EQ,ANIF(TERMAREA)
002780 D201 4002 6000 00002 00000 3672          MVC   0(2,CONSTCNT),0(WORK3)
002702 47F0 C2F0              02A66 3673          B     PASINB
                                  3674          ENDIF
                                  3676          CAR   2(WORK3),LOADR
002784 4100 0004              00004 3681          LA    RETURNPT,4(RETURNPT)
002788 D201 C102 6000 02918 00000 3682          MVC   IMBEFF(2),0(RETURNPT)
                                  3683 IMB1     PUSH  2(LOADR)
002784 D201 C2E4 3000 02A70 00000 3691          MVC   CASE1(2),0(LOADR)
002800 D201 805A 8052 0005A 00052 3692          MVC   ATENT1(2),ANIF(TERMAREA)
                                  3693          CAR   LISTSI,WORK3
                                  3698 IMB2     PUSH  2(WORK3)
002820 D201 C2E6 6000 02A6E 00000 3706          MVC   CASE(2),0(WORK3)
                                  3707          CAR   0(WORK3),WORK3
002846 D201 80E4 6000 00004 00000 3712          MVC   ATENT(2),0(WORK3)
                                  3713          CAR   2(WORK3),WORK3
002850 D201 80E6 6000 00006 00000 3718          MVC   ATENT+2(2),0(WORK3)
                                  3719          CAR   CASE1,WORK3
                                  3724 IMB3     PUSH  2(WORK3)
                                  3732          CAR   0(WORK3),WORK3
002890 D201 8004 6000 00004 00000 3737          MVC   POLY(2),0(WORK3)
002842 D201 8006 60E4 00006 00004 3738          MVC   POLY1(2),ATENT
                                  3739          CAR   2(WORK3),WORK3
002888 D201 8004 6000 00004 00000 3744          MVC   SAVE1(2),0(WORK3)
                                  3745          CALL  &AND
002802 D201 8052 8052 00052 00052 3748          MVC   COND(2),ANIF(TERMAREA)
                                  3749          CONS  POLYNOM,COND
002804 D201 8004 80C4 00004 00004 3766          MVC   POLY(2),SAVE1
002810 D201 8006 80E6 00006 00006 3767          MVC   POLY1(2),ATENT+2
002816 05FA                       3768          BALR  SYSLINK,SYSERROS
002818                            3769 IMBEFF   DS    H
                                  3770          SNOC  POLYNOM,COND
                                  3794          CONS  COND,ATENT1
                                  3811          PULL  SUITE
                                  3815          IF    H,SUITE,NE,ANIF(TERMAREA)
                                  3819          CAR   SUITE,WORK3
002BF0 47F0 C0E0              02872 3824          B     IMB3
                                  3825          ENDIF
                                  3827          CALLFCL CASE
                                  3831          PULL  SUITE
                                  3835          IF    H,SUITE,NE,ANIF(TERMAREA)
                                  3839          CAR   SUITE,WORK3
002A12 47F0 C030              02916 3844          B     IMB2
                                  3845          ENDIF
002A16 D201 80CC 80EA 000CC 000EA 3847          MVC   LIST(2),LISTSI
                                  3848          CALL  &RECURE
002A20 D201 80EA 80E8 000EA 000E8 3851          MVC   LISTSI(2),ATENT1
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                    3852              PULL  SUITE
                                    3856              IF    H,SUITE,NE,ANIF(TERMAREA)
                                    3850              CAR   SUITE,LCADR
002A4C 47F0 C05A              027E0 3865              B     INB1
                                    3866              ENDIF
002A50 4170 0004              00004 3858              LA    MWORK,4
002A54 1B07                         3869              SR    RETURNET,MWORK
002A56 D201 807C 4000 0C00C 00000   3870              MVC   LIST(2),0(CONSICNT)
                                    3871              CALL  FRECUPE
002A60 D201 4000 80EA C0000 000EA   3874              MVC   0(2,CONSICNT),LISTSI
002A66 4144 0002              00002 3875 PASINB       LA    CONSICNT,2(CONSICNT)
                                    3876              EXIT
002A6E                              3878 CASE          DS    H
002A70                              3879 CASE1         DS    H
                                    3880              RETURN
```

```
  LOC  OBJECT CODE      ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                     1890          ROUTINE &MIXTE
                                     1893          DECAF 1
                                     1896          IF     H,0(CONSICNT),EQ,ANIF(TERMAREA)
                                     1900          EXIT
                                     1902          ENDIF
                                     1904          IF     H,2(CONSICNT),EQ,ANIF(TERMAREA)
                                     1908          EXIT
                                     1910          ENDIF
001330 4133 0004             00004   1912          LA     RETURNPT,4(RETURNPT)
001340 D201 C074 9000 01394 00000    1913          MVC    MIXEFF(2),0(RETURNPT)
                                     1914          CAR    0(CONSICNT),WORK1
001358 D201 8000 4002 00000 00002    1919 MIX1     MVC    POLY1(2),2(CONSICNT)
00135E 4144 0002             00002   1920          LA     CONSICNT,2(CONSICNT)
001362 D201 80E8 1002 000E8 00002    1921          MVC    ATENT1(2),2(WORK1)
                                     1922          CAR    0(WORK1),WORK1
                                     1927          CAR    2(WORK1),WORK1
001380 D201 800A 1000 0000A 00000    1932          MVC    POLY(2),0(WORK1)
001386 5010 80E4             000E4   1933          ST     WORK1,ATENT
00138A 057A                          1934          BALR   SYSLINK,SYSREGS
001394                               1935 MIXEFF   DS     H
001396 5810 80E4             000E4   1936          L      WORK1,ATENT
                                     1937          CALLRECU 0(WORK1)
0013A2 D201 1000 800E 00000 0000E    1941          MVC    0(2,WORK1),POLYNOM
                                     1942          IF     H,ATENT1,NE,ANIF(TERMAREA)
                                     1946          CAR    ATENT1,WORK1
                                     1951          DECAF 1
0013C8 4750 C035             01358   1954          B      MIX1
                                     1955          ENDIF
0013CC 4170 0004             00004   1957          LA     MWORK,4
0013D0 1B37                          1958          SR     RETURNPT,MWORK
                                     1959          CALLRECU 0(CONSICNT)
                                     1963          RETURN
```

```
  LOC   OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                     2323          ROUTINE BRANCH
                                     2326          DECAR 1
                                     2329          CAR    2(CONSICNT),WORK1
                                     2334          IF     H,2(WORK1),EQ,ANIF(TERMAREA)
                                     2338          CAR    0(WORK1),WORK1
                                     2343          IF     H,2(WORK1),EQ,ANIF(TERMAREA)
                                     2347          CAR    0(WORK1),WORK1
                                     2352          IF     H,2(WORK1),EQ,ANIF(TERMAREA)
                                     2356          CAR    0(WORK1),WORK2
                                     2361          IF     C,1(WORK2),EQ,X'5E'
                                     2364          CAR    2(WORK2),WORK2
                                     2369          CAR    2(WORK2),WORK2
                                     2374          CAR    2(WORK2),WORK2
                                     2379          CAR    2(WORK2),WORK2
                                     2384          IF     H,0(WORK2),LT,0
                                     2388          CAR    0(WORK2),WORK3
                                     2393          WHILE H,2(WORK3),NE,ANIF(TERMAREA)
                                     2398 _        CAR    2(WORK3),WORK3
                                     2403          ENDWHILE
001904 41FF 0002              00002  2406          LA     WORK3,2(WORK3)
                                     2407          ENDIF ELSE
00190C 18F2                          2410          LR     WORK3,WORK2
                                     2411          ENDELSE
00190E D201 F000 4000 00000 00000    2413          MVC    0(2,WORK3),0(CONSICNT)
                                     2414          CAR    0(WORK3),WORK3
001924 D201 F000 F002 00000 00002    2419          MVC    0(2,WORK3),2(WORK3)
00192A D201 F002 2002 00002 00002    2420          MVC    2(2,WORK3),2(WORK2)
001930 D201 2002 8052 00002 00052    2421          MVC    2(2,WORK2),ANIF(TERMAREA)
                                     2422          CAR    2(WORK3),WORK2
                                     2427          IF     H,0(WORK2),NE,VIDE
001952 D201 F002 2000 00002 00000    2431          MVC    2(2,WORK3),0(WORK2)
                                     2432          ENDIF ELSE
00195C D201 F002 8052 00002 00052    2435          MVC    2(2,WORK3),ANIF(TERMAREA)
                                     2436          ENDELSE
001962 D201 4000 1000 00000 00000    2438          MVC    0(2,CONSICNT),0(WORK1)
001968 D201 1000 80E4 00000 000E4    2439          MVC    0(2,WORK1),VIDE
                                     2440          CALLFECL 2(CONSICNT)
001976 47F0 C1??              019?4  2444          B      TERMINE
                                     2445          ENDIF
                                     2447          ENDIF
                                     2449          ENDIF
                                     2451          ENDIF
0019?A D201 80CE 4000 000CE 00000    2453          MVC    LIST1(2),0(CONSICNT)
                                     2454          SNOC   2(CONSICNT),LIST1
0019DE D201 4000 80CE 00000 000CE    2478          MVC    0(2,CONSICNT),LIST1
                                     2479 TERMINE  RETURN
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                   2880          ROUTINE &SUITDIV
                                   2883 *
                                   2884          DECAF 1
                                   2887          IF    H,0(CONSICNT),LT,0
                                   2891          CAR   0(CONSICNT),WORK3
                                   2896          ENDIF ELSE
001572 1874                        2899          LR    WORK3,CONSICNT
                                   2900          ENDELSE
                                   2902          IF    H,2(CONSICNT),LT,0
                                   2906          CAR   2(CONSICNT),WORK2
                                   2911          ENDIF ELSE
001574 4124 1002           00002   2914          LA    WORK2,2(CONSICNT)
                                   2915          ENDELSE
                                   2917          IF    H,0(WORK2),NE,CORST
                                   2921          IF    H,(WORK3),NE,CORST
001580 D201 8000 8000 00000 00152  2925          MVC   LIST(2),ANIF(TERMAREA)
                                   2926          CONS  0(CONSICNT),LIST
                                   2943          CONS  2(CONSICNT),LIST
                                   2960          DECAF 2
                                   2965          CONS  LIST,0(CONSICNT)
001504 4144 1002           00002   2982          LA    CONSICNT,2(CONSICNT)
                                   2983          EXIT
                                   2985          ENDIF
001500 D201 8000 4002 00000 00002  2987          MVC   POLY1(2),2(CONSICNT)
001502 D201 8005 5002 0005E 00002  2988          MVC   LIST1(2),2(WORK3)
001500 4144 1002           00002   2989          LA    CONSICNT,2(CONSICNT)
                                   2990          WHILE H,LIST1,NE,ANIF(TERMAREA)
                                   2995          CAR   LIST1,WORK2
001503 D201 8005 2002 0005E 00002  3010          MVC   LIST1(2),2(WORK2)
                                   3031          CAR   0(WORK2),WORK1
                                   3036          CAR   2(WORK1),WORK1
001500 D201 8004 1000 00004 00000  3011          MVC   POLY(2),0(WORK1)
001504 5010 8054           00054   3012          ST    WORK1,ATENT
                                   3013          CALL  &CONSDIV
001500 5810 8054           00054   3016          L     WORK1,ATENT
001505 D201 1000 8005 00000 0005E  3017          MVC   0(2,WORK1),POLYNEW
                                   3018          CALLSECL POLY
                                   3022          ENDWHILE
                                   3025          CALLSECU POLY1
                                   3029          DECAF 5
                                   3040          CAR   0(CONSICNT),WORK1
                                   3045          CONS  2(WORK1),0(CONSICNT)
002001 4144 1006           00006   3062          LA    CONSICNT,6(CONSICNT)
                                   3063          ENDIF ELSE
                                   3066          IF    H,0(WORK3),NE,CORST
002004 D201 8004 4000 00004 00000  3070          MVC   POLY(2),0(CONSICNT)
002005 D201 8005 2002 0005E 00002  3071          MVC   LIST1(2),2(WORK2)
002000 4144 1002           00002   3072          LA    CONSICNT,2(CONSICNT)
                                   3073          WHILE H,LIST1,NE,ANIF(TERMAREA)
                                   3078          CAR   LIST1,WORK2
002050 D201 8005 2002 0005E 00002  3083          MVC   LIST1(2),2(WORK2)
                                   3084          CAR   0(WORK2),WORK1
                                   3089          CAR   2(WORK1),WORK1
002024 D201 8000 1000 0000E 00000  3094          MVC   POLY1(2),0(WORK1)
```

```
FILE KECT                                                   CAMBRIDGE  SCIENTIFIC  CENTER

  LOC  OBJECT CODE      ADDR1 ADDR2  STMT   SOURCE STATEMENT

0020EC 5010 8054            00054  3095          ST    WORK1,ATENT
                                   3096          CALL  ECONSDIV
0020F4 5810 8054            00054  3099          L     WORK1,ATENT
0020F8 D201 1000 803E 00000 0003E  3100          MVC   0(2,WORK1),POLYNOM
                                   3101          CALLRECL POLY1
                                   3105          ENDWHILE
                                   3108          CALLRECU POLY
                                   3112          DECAR 5
                                   3123          CAR   10(CONSICNT),WORK1
                                   3128          CONS  2(WORK1),0(CONSICNT)
002168 4144 0006            00006  3145          LA    CONSICNT,6(CONSICNT)
                                   3146          ENDIF ELSE
                                   3149          APPLIQ ECONSDIV
002178 4199 0004            00004  3152          LA    RETURNPT,4(RETURNPT)
                                   3153          APPLIQ ECEPOLY
002184 4170 0004            00004  3156          LA    MWORK,4
002188 1B97                        3157          SR    RETURNPT,MWORK
00218A D203 0004 3052 00004 00052  3158          MVC   4(CONSICNT),ANIF(TERMAREA)
                                   3159          CONS  2(WORK2),4(CONSICNT)
                                   3176          CONS  0(WORK3),4(CONSICNT)
002214 4144 0006            00006  3193          LA    CONSICNT,6(CONSICNT)
                                   3194          CALL  EIMBRIC
                                   3197          DECAR 7
                                   3212          CONS  12(CONSICNT),0(CONSICNT)
                                   3229          ENDELSE
00227A 4144 0006            00006  3231          LA    CONSICNT,6(CONSICNT)
                                   3232          ENDELSE
                                   3234          RETURN
```

```
 LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                   3238          ROUTINE TRITUR
                                   3241 *
002282 4100 0004                   00004  3242          LA    RETURNPT,4(RETURNPT)
002286 D201 C184 9000 02402 00000  3243          MVC   TRITURE(2),0(RETURNPT)
                                   3244          DECAP 2
                                   3249          IF    H,2(CONSTCNT),EQ,ANIF(TERMAREA)
                                   3253          IF    H,4(CONSTCNT),EQ,ANIF(TERMAREA)
002240 41F0 0004                   00004  3257          LA    PARMADD,10
                                   3258          EXIT
                                   3260          ENDIF ELSE
002288 47F0 C114                   02390  3263          B     TRIT1
                                   3264          ENDELSE
                                   3266          ENDIF ELSE
                                   3269          IF    H,4(CONSTCNT),EQ,ANIF(TERMAREA)
                                   3273          CONS  2(CONSTCNT),2(CONSTCNT)
                                   3290          ENDIF ELSE
002312 D201 8005 8052 00005 00052  3293          MVC   POLYNOM(2),ANIF(TERMAREA)
                                   3294          CONS  2(CONSTCNT),POLYNOM
                                   3311          CONS  POLYNOM,4(CONSTCNT)
002330 D201 8005 4004 00005 00004  3329 TRIT1    MVC   LIST1(2),4(CONSTCNT)
                                   3329          PUSH  LIST1
002350 D201 8005 8052 00005 00052  3337          MVC   POLYNOM(2),ANIF(TERMAREA)
                                   3338          WHILE H,LIST1,NE,ANIF(TERMAREA)
                                   3343          CAR   LIST1,WORK1
002370 D201 8005 1002 00005 00002  3348          MVC   LIST1(2),2(WORK1)
                                   3349          PUSH  POLYNOM
002388 D201 8010 800E 00010 0000E  3357          MVC   POLY1(2),POLYNOM
002404 D201 801A 1000 0001A 00000  3358          MVC   POLY(2),0(WORK1)
00240A 0571                        3359          BALR  SYSLINK,SYSPROCS
00240C                             3360 TRITURE  DS    H
                                   3361          PULL  POLY
                                   3365          CALLPROC POLY
                                   3369          ENDWHILE
                                   3372          PULL  LIST1
                                   3376          ACCROCHE 0(CONSTCNT),POLYNOM
                                   3388          CALLPROC LIST1
002450 D201 4004 8002 00004 00002  3392          MVC   4(2,CONSTCNT),LIMITE
                                   3393          ENDELSE
                                   3395          ENDELSE
002462 4170 0004                   00004  3397          LA    WORK,4
002466 1B87                        3398          SR    RETURNPT,WORK
                                   3399          RETURN
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                   3403          ROUTINE &VOIRMIN
                                   3406 *
                                   3407          IF    H,0(CONSTCNT),GT,0
002478 D201 8000 8052 00000 00052  3411          MVC   LIST(2),ANIF(TERMAREA)
                                   3412          CONS  0(CONSTCNT),LIST
002480 9233 4000       00000       3429          MVI   0(CONSTCNT),X'3E'
002484 9233 4001       00001       3430          MVI   1(CONSTCNT),X'33'
                                   3431          CONS  0(CONSTCNT),LIST
                                   3448          DECAF 1
                                   3451          CONS  LIST,0(CONSTCNT)
                                   3468          ENDIF ELSE
                                   3471          IF    H,0(CONSTCNT),LT,LIMITS
002560 D201 8000 8052 00000 00052  3475          MVC   LIST(2),ANIF(TERMAREA)
                                   3476          CONS  0(CONSTCNT),LIST
                                   3493          CONS  1,LIST
002568 9233 4000       00000       3511          MVI   0(CONSTCNT),X'3E'
00256C 9233 4001       00001       3512          MVI   1(CONSTCNT),X'33'
                                   3513          CONS  0(CONSTCNT),LIST
                                   3530          DECAF 1
                                   3533          CONS  LIST,0(CONSTCNT)
                                   3550          ENDIF ELSE
                                   3553          CAR   0(CONSTCNT),WORK1
                                   3558          IF    H,0(WORK1),EQ,CODST
                                   3562 VOIR1    CAR   2(WORK1),WORK1
                                   3567          CAR   0(WORK1),WORK2
                                   3572          CAR   2(WORK2),WORK2
002600 D201 8000 2000 00000 00000  3577          MVC   LIST(2),0(WORK2)
                                   3578          CALL  &CHSIGN
002606 D201 2000 8000 00000 00000  3581          MVC   0(2,WORK2),LIST
                                   3582          IF    H,2(WORK1),NE,ANIF(TERMAREA)
002678 4750 C230               02690  3586          B     VOIR1
                                   3587          ENDIF
                                   3589          CAR   0(CONSTCNT),WORK1
                                   3594          DECAF 2
                                   3599          CONS  2(WORK1),0(CONSTCNT)
002746 4144 0002               00002  3616          LA    CONSTCNT,2(CONSTCNT)
                                   3617          ENDIF ELSE
002748 D201 8000 4000 00000 00000  3620          MVC   LIST(2),0(CONSTCNT)
                                   3621          CALL  &CHSIGN
                                   3624          DECAF 1
                                   3627          ACCROCHE 0(CONSTCNT),LIST
                                   3639          ENDELSE
                                   3641          ENDELSE
                                   3643          ENDELSE
                                   3645          RETURN
```